WRDC-TR-90-8007
Volume V
Part 9
Section 3 of 5

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 9 - Neutral Data Manipulation Language (NDML) Precompiler Development Specification
Section 3 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

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
September 1990

3. REPORT TYPE AND DATES COVERED
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 3 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 3 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

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std Z239-18
298-102

SECTION 21

FUNCTION CDCE - Generate CS to ES runtime code

This function generates COBOL source code according to the ANSI X3.23-1974 standard into the Conceptual Schema to External Schema transform program.

This code generates the interface and calls to user defined complex mapping algorithms. In the cases where no complex mapping algorithms are defined for conceptual fields, simple moves are generated to the corresponding external fields and null flags.

21.1 Inputs

1. WORK-FILE-1 PIC X(30)

   Contains the name of the file into which working storage statements may be generated.

2. WORK-FILE-2 PIC X(30)

   Contains the name of the file into which procedure division statements will be generated.

3. STRAIGHT-MOVE-FLAG PIC X

   Indicates whether the destination fields are external fields or working storage fields.

4. CS-ACTION-LIST included in CSAL copy member

   Contains conceptual representation of fields to be retrieved.

5. ES-ACTION-LIST included in ESAL copy member

   Contains external representation of fields to be retrieved.

6. TARGET-HOST PIC XXX

   Host upon which the CS-ES transform program will execute at runtime.

7. CMA-FLAG PIC 9

If zero, don't use CMA logic. If non-zero, use CMA logic.

21.2 CDM Requirements

ENTITY CLASS

COMPLEX_MAPPING_PARM
MODULE_PARAMETER
USER_DEF_DATA_TYPE

21.3 Internal Requirements

None

Macro Generation

Macros are code templates with optional substitutable parameters which allow the generated code to be more independent of the generating programs. All macros are to be generated through calls to CDMACR. This routine requires the following parameters:

Input

FILE-NAME	PIC X(30)	included in MACDAT copy member
LIBRARY-NAME	PIC X(30)	included in MACDAT copy member
MACRO-NAME	PIC X(8)	included in MACDAT copy member
SUBSTITUTION-LIST		included in SBSTLST copy member

Output

RET-STATUS	PIC X(5)

FILE-NAME contains the name of the file to which code is to be generated. This file must be closed prior to the CDMACR call. Upon return to CDCE, FILE-NAME must be reopened for EXTEND to allow code to be generated at the end of the file.

LIBRARY-NAME contains the name of the host upon which the generated code will execute at runtime. This value is identical to the CDCE input parameter TARGET-HOST.

MACRO-NAME contains the name of the macro to be generated, for example "CDCE01".

SUBSTITUTION-LIST is described by the following structure:

```
01 SUBSTITUTION-LIST
   03    SL-USED     PIC 99
   03    SL-MAX      PIC 99
   03    SL-ROW-SIZE     PIC 99
   03    SL-ENTRY OCCURS 8 TIMES
           INDEXED BY SL-INDEX
   05    SL-PARAMETER     PIC X(30)
   05    SL-SUBSTVAL      PIC X(30)
```

SUBSTITUTION-LIST is populated be setting the SL-USED to the number of parameter values the macro requires. SL-PARAMETER (index) contains the macro parameter to be substituted for, for example P1. SL-SUBST-VAL (index) contains the corresponding substitution value, for example CS-NDML-NO.

RET-STATUS is a value equal to KES-SUCCESSFUL as defined in the ERRCDM copy member, if the macro generation was successful.

21.4 Processing

1. Open EXTEND WORK-FILE1, WORK-FILE2.

2. If CMA-FLAG equals zero, perform CASE1 for each CS-ACTION-LIST entry (step 6).

3. Parse SQL statement 1.

```
SELECT MOD_ID, PARM_ID, CONSTANT_VALUE, UNION_DISC
FROM COMPLEX_MAPPING_PARM
WHERE ALG_USE_CODE = "R"  AND MOD_ID IN
(SELECT MOD_ID FROM COMPLEX_MAPPING_PARM
WHERE TAG_NO = :TAG-NO-WS) AND MOD_INST IN
(SELECT MOD_INST FROM COMPLEX_MAPPING_PARM
WHERE TAG_NO = :TAG-NO-WS) AND MOD_ID IN
(SELECT MOD_ID FROM COMPLEX_MAPPING_PARM
WHERE DI_NO = :DI-NO-WS) AND MOD_INST IN
(SELECT MOD_INST FROM COMPLEX_MAPPING_PARM
```


```
WHERE DI_NO = :DI-NO-WS)

ORDER BY PARM_ID
```

4. If CMA-FLAG not equal zero, perform the following steps for each CS-ACTION-LIST entry, where the CS-ES-PTR is not equal to zero.

   4.1 Move the current CS-AUC to the tag number parameter (TAG-NO-WS) in SQL Statement 1.

   4.2 Move the current ES-DI-NO, pointed to by the current CS-ES-PTR, to the data item number parameter (DI-NO-WS) in SQL Statement 1.

   4.3 Open cursor 1.

   4.4 Fetch the first row of data

      4.4.1 If no rows are retrieved, perform CASE1 for the current CS-ACTION-LIST entry (step 6).

      4.4.2 If rows are retrieved, perform CASE2 for the current CS-ACTION-LIST entry (step 7).

   4.5 Close cursor 1.

5. Close WORK-FILE1, WORK-FILE2.

6. CASE1 - No complex CS-ES mapping is defined for the current AUC-data item combination.

   6.1 If the CS-ACTION-LIST entry is not to be projected and STRAIGHT-MOVE-FLAG equals "Y", exit from CASE1.

   6.2 Generate in WORK-FILE2, conceptual moves using the CDCE03 macro with the following parameter substitutions:

PARAMETER	SUBSTITUTION VALUE
P1	If STRAIGHT-MOVE-FLAG = Y, substitute the character string "ES". If STRAIGHT-MOVE-FLAG = "N", substitute the character string "WS".

P2	Current CS-INDEX value
P3	Current CS-ES-PTR value
P4	Current CS-NDML-NO value

7. CASE2 - A complex CS-ES mapping is defined for the current AUC-data item combination.

7.1 If the CS-ACTION-LIST entry is not to be projected and STRAIGHT-MOVE-FLAG equals "Y", exit from CASE2.

7.2 Generate in WORK-FILE2, the null flag test using the CDCE01 macro with the following parameter substitutions:

PARAMETER	SUBSTITUTION VALUE
P1	Current CS-INDEX
P2	If STRAIGHT-MOVE-FLAG = "Y", substitute the character string "ES", otherwise substitute the character string "WS".
P3	Current CS-ES-PTR value
P4	Current CS-NDML-NO value

7.3 Process the row returned from SQL statement 1 as follows:

7.3.1 Execute SQL statement 2:

```
SELECT A.TYPE_ID,
       A.MAX_SIZE,
       A.NO_OF_DECIMALS,
       A.DATA_TYPE_NAME

INTO

       :TYPE-ID-WS,
       :MAX-SIZE-WS,
       :NO-OF-DEC-WS,
       :DATA-TYPE-NAME-WS

FROM
```

```
USER_DEF_DATA_TYPE A,
MODULE_PARAMETER B

WHERE

((B.MOD_ID = :MOD-ID-WS AND
  B.PARM_ID = :PARM-ID-WS) AND
 (B.DATA_TYPE_NAME = A.DATA_TYPE_NAME))
```

Using the module name returned in SQL statement 1 for parameter 1 (MOD-ID-WS) and the parameter number for parameter 2 (PARM-ID-WS).

7.3.2 Send CDPIC the returned type, maximum size and number of decimals as input parameters to generate a picture clause.

7.3.3 Generate the following working storage element in WORK-FILE1.

```
01  CS-ES-VAR-csindex-parmid    pic-clause.
```

Where csindex is the current CS-INDEX value and parmid is the current PARM-ID value returned from the CDM. Pic-clause was generated in 7.3.2.

7.3.4 If the current union discriminator returned (UNION-DISC-WS) from the CDM is equal to "1" (indicating a tag) generate the following Procedure Division move of the CS variable to the user module parameter in WORK-FILE2.

```
MOVE CS-VAR-csindex TO
CS-ES-VAR-csindex-parmid.
```

csindex is the current CS-INDEX value and parmid is the current PARM-ID value returned from the CDM.

7.3.5 If the union discriminator returned (UNION-DISC-WS) from the CDM is equal to "2" (indicating a constant).

7.3.5.1 If the constant is a character (type retrieved from CDM equals "C") generate the following Procedure Division move of the constant value

to the user module parameter in WORK-FILE2.

MOVE "constant-value" TO CS-ES-VAR-csindex-parmid

7.3.5.2 If the constant is numeric (type retrieved from the CDM not equal "C") generate the following Procedure Division move of the constant value to the user module parameter in WORK-FILE2.

MOVE constant-value TO CS-ES-VAR-esindex-parmid

7.3.5.3 In both 7.3.5.1 and 7.3.5.2, constant-value is the current CONSTANT-VALUE returned from the CDM, csindex is the current CS-INDEX value and parmid is the current PARM-ID returned from the CDM.

7.3.6 If the union discriminator returned (UNION-DISC-WS) from the CDM is equal "5" retain the current PARM-ID value for parameter 6 in the macro generation of step 7.7.

7.3.7 Fetch the next SQL statement 1 row. Keep a count of the number of rows retrieved (i.e. number of parameters defined for the user module).

If another row is returned, continue processing at step 7.3.

7.4. Generate on WORK-FILE2, the user module call statement.

CALL "mod-id" USING

where mod-id is the current MOD-ID value returned from the CDM.

7.5 Generate on WORK-FILE2, the module parameter list by generating one parameter for each PARM-ID fetched, starting at 1.

CS-ES-VAR-csindex-1
.
.
.
CS-ES-VAR-csindex-parm-counter

where csindex equals the current CS-INDEX value and parm-counter is the total number of rows retrieved (i.e. maximum number of parameters for the user module) as calculated in step 7.3.7.

7.6 Generate on WORK-FILE2, the status parameter and the terminating period.

RET-STATUS.

7.7 Generate on WORK-FILE2, the user module status checking logic and the move from the user module return parameter, using the CDCE02 macro with the following parameter substitutions:

PARAMETER	SUBSTITUTION VALUE
P1	current CDM MOD-ID value
P2	current CS-INDEX value
P3	If STRAIGNT-MOVE-FLAG equals "Y", substitute the character string "ES", otherwise substitute the character string "WS".
P4	current CS-NDML-NO value
P5	current CS-ES-PTR value
P6	PARM-ID value retained in step 7.3.6.

21.5 Output

1. RET-STATUS PIC X(5)

Error status which will equal KES-SUCCESSFUL as defined in the ERRCDM copy member if CDCE runs successfully.

SECTION 22

FUNCTION PRE9.1 REQUEST PROCESSOR GENERATOR SUPPORT FUNCTIONS

This document will address the design of support routines used by the Request Processor Generators within the IISS Precompiler architecture. These support routines will be used when generating code to perform the conceptual to internal transformation of NDML query search parameters and the internal to conceptual transformation of retrieved data fields. Included in this document is a design specification for each support routine as identified below:

CDCI	Generate Conceptual/Internal Transformation
CDCMD	Retrieve Conceptual Meta Data
CDCWF	Combine Generator Work Files
CDIC	Generate Internal/Conceptual Transformation
CDIMD	Retrieve Internal Meta Data
CDMSG	Generate Coneptual Schema Search Parameters
CDPIC	Generate COBOL Picture Clause
CDPRM	Generate Internal Schema Search Parameters
CDQDF	Generate Internal Schema Retrieval Qualification Variables
CDRDF	Generate Internal Schema Retrieval Data Fields
CDRFT	Generate Conceptual Schema Retrieval Data Fields
CDMACR	Macro Expander
CDCMPRM	Generate Complex Mapping Algorithm Parameters
CDGENRT	Generate A COBOL Record Layout For A Specified Record Type
CDGENIF	Generate COBOL IF Statement
CDGTV	Generate Tag Variable Definitions
CDGDF	Generate Datafield and Indicator Variables
CDGNV	Generate User-Defined NULL Variable Names
CDRPCIF	Generate COBOL IF Statement for Conceptual Schema
CDRPIIF	Generate COBOL IF Statement for Internal Schema
CDRPUIF	Generate COBOL IF Statement for Union Discriminator for Specified Record Type

Each design specification will include routine inputs and outputs and a processing description. This document describes existing functions which are used by the Request Processor Generators (PRE9.2, PRE9.3, PRE9.4, PRE9.5) and are mentioned by name in the following documentation on the RP Generators or in the code for each RP Generator.

22.1 CDCI Generate Conceptual/Internal Transformation

This routine will generate ANSI X3.23-1974 COBOL Procedure Division Code required for the transformation of search/update parameters from Conceptual Schema Format to Internal Schema Format into a SQL-based Request Processor Subroutine.

22.1.1 Inputs

1. WORK-FILE PIC X(30)

   WORK-FILE contains the name of the file where the COBOL statements will be generated.

2. SUBTRANS-ID PIC 999

   SUBTRANS-ID contains the subtransaction identifier.

3. IS-ACTION-LIST in ISAL copy member of IISSCLIB

   IS-ACTION-LIST contains the Internal Schema Representation of the data items to be retrieved or updated.

4. IS-QUALIFY-LIST in ISQUAL copy member of IISSCLIB

   IS-QUALIFY-LIST contains the Internal Schema Representation of the WHERE clause.

5. CMA-TABLE in copy member of IISSCLIB

   CMA-TABLE contains complex mapping Algorithm parameter information.

6. NUMERIC-NULL PIC X(30)

   NUMERIC-NULL contains the user specified numeric null value or NULL to indicate that the database null value is to be used.

7. CHAR-NULL PIC X(30)

   CHAR-NULL contains the user specified character null value or NULL to indicate that the database null value is to be used.

22.1.2 CDM Requirements

None

22.1.3 Internal Requirements

None

22.1.4 Macro Generation

Macros are code templates with optional substitutable parameters which allow generated codes to be more independent of the generating programs. All macros are to be generated through calls to CDMACR. This routine requires the following parameters:

Input

```
FILE-NAME          PIC X(30)      included in MACDAT copy member
LIBRARY-NAME       PIC X(30)      included in MACDAT copy member
MACRO-NAME         PIC X(8)       included in MACDAT copy member
SUBSTITUTION-LIST                 included in SBSTLST copy member
```

Output

```
RET-STATUS         PIC X(5)
```

FILE-NAME contains the name of the file to which code is generated. This file must be closed prior to the CDMACR call. Upon return to CDCI, FILE-NAME must be reopened for EXTEND to allow code to be generated at the end of the file.

LIBRARY-NAME contains the name of the library containing CDCI macros. All CDCI macros will have a library name of SQL.

MACRO-NAME contains the name of the macro to be generated, for example: CDCI01.

SUBSTITUTION-LIST is described by the following structure:

```
01  SUBSTITUTION-LIST.
    03  SL-USED              PIC 99.
    03  SL-MAX               PIC 99.
    03  SL-ROW-SIZE          PIC 99.
    03  SL-ENTRY OCCURS 8 TIMES
        INDEXED BY SL-INDEX.
        05  SL-PARAMETER     PIC X(30).
        05  SL-SUBST-VAL     PIC X(30).
```

SUBSTITUTION-LIST is populated by setting SL-USED to the number of parameter values the macro requires. SL-PARAMETER (index) contains the macro parameter to be substituted, e.g. P1. SL-SUBST-VAL (index) contains the corresponding substitution value, e.g. IS-DFNO.

22.1.5 Processing

1. Open EXTEND WORK-FILE.

2. Move KES-SUCCESSFUL to RET-STATUS.

3. Process IS-ACTION-LIST for insert and modify.

   3.1 If the NDML Statement is not an INSERT or a MODIFY (IS-ACTION not equal "I" or "M"), go to step 4.

   3.2 Mark the IS-ACTION entries to be transformed.

   Scan the IS-ACTION-LIST searching for used IS-SUBTRANS-IDs which match the input parameter SUBTRANS-ID and which have non-zero IS-CS-PTRs. For each entry satisfying the above requirements, set the corresponding IS-FLAG to "1".

   3.3 Transform each marked IS-ACTION entry.

   For each used IS-ACTION entry which has IS-FLAG equal "1", perform the following steps. When all marked entries have been transformed, go to step 3.4.

   3.3.1 If the current IS-MAP-ALG-PTR equals zero, generate the following MOVE statements, else go to step 3.3.3.

   MOVE MSGI-VAR-isindex TO IS-VAR-isindex.

   MOVE ZERO TO INDP-isindex.

   where isindex is the current 3-digit IS-INDEX value.

   3.3.2 Set the current IS-FLAG to zero and go to step 3.3.

   3.3.3 A complex mapping algorithm has been defined for the current IS-ACTION entry. Perform the following steps:

   3.3.3.1 Search the CMA-TABLE for tags to be transformed.

   Scan the CMA-TAG-NO entries pointed to by the current IS-ALG-PTR looking for a non-zero tag number. If no more non-zero tag numbers are found (at least "1" will be found), continue with step 3.3.3.2. If found, search the IS-ACTION-LIST looking for an entry with IS-ALG-ID equal to the current CMA-MOD-ID and IS-PARM-NO equal to the current CMA-PARM-NO and

IS-ALG-PTR equal to the current CMA-ALG-ENTRY index.

When found, generate the following MOVE statement:

MOVE MSGI-VAR-isindex TO PARM-mod-inst-pno.

where isindex is the current 3-digit IS-INDEX value, mod is the current CMA-MOD-ID value, inst is the current CMA-MOD-INST value and pno is the current CMA-PARM-NO value.

Set the current IS-FLAG to zero.

Go back to step 3.3.3.1. and search for another tag to be transformed.

3.3.3.2 Generate MOVE statements for complex mapping algorithm constants, if any.

Scan the current CMA-ALG-ENTRY searching for a CMA-PARM-ENTRY which has CMA-TAG-NO, CMA-RT-NO and CMA-DF-NO equal zero.

For each such entry found, generate one of the two following MOVE statements. When MOVE statements have been generated for all constants, continue at step 3.3.3.3.

If the CMA-PARM-TYPE equals "C", generate

MOVE "constval" TO PARM-mod-inst-pno.

Place the quote marks around the CMA-CONST-VAL to generate a character literal.

If the CMA-PARM-TYPE does not equal "C", generate

MOVE constval TO PARM-mod-inst-pno.

where constval is the current CMA-CONST-VAL value, mod is the current CMA-MOD-ID value, inst is the current CMA-MOD-INST value and pno is the current CMA-PARM-NO value.

3.3.3.3 Generate the CALL statement to the user module.

CALL "mod" USING

where mod is the current CMA-MOD-ID value.

3.3.3.4 Generate the parameter list. For each parameter in the current CMA-PARM-ENTRY, generate the following:

PARM-mod-inst-pno

where mod is the current CMA-MOD-ID value, inst is the current CMA-MOD-INST value and pno ranges in order over all CMA-PARM-NO values in the current CMA-ALG-ENTRY.

3.3.3.5 Generate the status parameter and terminating period.

RET-STATUS.

3.3.3.6 Generate the status checking logic by substituting the value of the current CMA-MOD-ID for parameter P1 in macro CDCI01.

3.3.3.7 Generate the output parameter MOVE statements. Search the current CMA-ALG-ENTRY for CMA-DF-NO elements not equal zero. For each non-zero CMA-DF-NO, generate:

MOVE PARM-mod-inst-pno To IS-mod-inst-pno.

MOVE ZERO TO INDP-mod-inst-pno.

where mod is the current CMA-MOD-ID value, inst is the current CMA-MOD-INST value and pno is the CMA-PARM-NO value of the CMA-PARM-ENTRY with a

non-zero CMA-DF-ID.

Go to step 3.3 to process the next IS-ACTION entry, if any.

3.4 Insert Nulls for any unmapped fields.

Scan the IS-ACTION-LIST searching for a used entry which has IS-MAPPED-TO-FLAG equal "N" and IS-SUBTRANS-ID equal SUBTRANS-ID. For each qualifying entry found, perform the following steps.

If the IS entry has a character data type and the database null value is to be used (IS-TYPE equals "C" and CHAR-NULL equals NULL), generate the following MOVE statement:

```
MOVE ZERO TO IS-VAR-isindex.
MOVE -1 TO INDP-isindex.
```

where isindex is the current IS-INDEX value.

If the IS entry has a character data type and a user specified null value is to be used (IS-TYPE equals "C" and CHAR-NULL not equal NULL), generate the following MOVE statements:

```
MOVE charnull TO IS-VAR-isindex.
MOVE ZERO TO INDP-isindex.
```

where charnull is the value contained in input parameter CHAR-NULL and isindex is the current IS-INDEX value.

If the IS entry has a numeric data type and the database null value is to be used (IS-TYPE not equal C and NUMERIC-NULL equal NULL), generate the following move statements.

```
MOVE ZERO TO IS-VAR-isindex.
MOVE -1 TO INDP-isindex.
```

where isindex is the current IS-INDEX value.

If the IS-ENTRY has a numeric data type and a user specified null value is to be used, generate the following MOVE statements:

```
MOVE numnull TO IS-VAR-isindex
MOVE ZERO TO INDP-isindex.
```

where numnull is the value contained in input parameter NUMERIC-NULL and isindex is the current IS-INDEX value.

When all unmapped fields have been processed, go to step 3.5.

3.5 Scan the IS-ACTION-LIST searching for a used entry with IS-SUBTRANS-ID equal SUBTRANS-ID and IS-CS-PTR equal zero. For each such entry, perform the following steps. When all such entries have been processed, go to step 4.

3.5.1 Generate the following statement.

MOVE ZERO TO INDP-isindex.

Where isindex is the current IS-INDEX value.

3.5.2 If the current IS-DATA-TYPE equal C, generate the following statement.

MOVE "uval" TO IS-VAR-isindex.

Where "uval" is the value of the current IS-UNION-VALUE and isindex is the current IS-INDEX value.

Place quotes around the union value to make it a character literal.

3.5.3 If the current IS-DATA-TYPE not equal C, generate the following statement.

MOVE uval TO IS-VAR-isindex.

where uval is the current IS-UNION-VALUE and isindex is the current IS-INDEX value.

4. Process IS-QUALIFY-LIST for select, Type-1 Referential Integrity Test, Type-2 Referential Integrity Test, Key Uniqueness Test, modify and delete.

4.1 If IS-ACTION equals "S" or "1" or "2" or "K" or "M" or "D", perform the following steps, otherwise go to step 5.

4.2 Scan the IS-QUALIFY-LIST searching for entries which have ISQ-TYPE equal "2" and ISQ-SUBTRANS-IDL equal the input parameter

SUBTRANS-ID and ISQ-CSQ-PTR not equal zero and ISQ-OP not equal "NL" and ISQ-OP not equal "NN" and ISQ-EVAL-FLAG not equal zero.

For each entry satisfying the above requirements, generate the following statement.

MOVE MSG-VAR-isqindex TO ISQL-VAR-isqindex.

where isqindex is the 3-digit ISQ-INDEX.

5. Close WORK-FILE and terminate processing.

22.1.6 Outputs

1. RET-STATUS PIC X(5)

RET-STATUS contains the CDCI completion status. A value equal to KES-SUCCESSFUL as defined in the ERRCDM copy member indicates success.

CDCI MACRO

Library: SQL

Macro: CDCI01

```
If RET-STATUS NOT = KES-SUCCESSFUL
    STRING "P1" DELIMITED BY SPACE
    "TRANSFORM PROGRAM FAILED"
    DELIMITED BY SIZE INTO MESG-DESC
MOVE RET-STATUS TO RP-STATUS
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
```

22.2 CDCMD Retrieve Conceptual Meta Data

This routine will use a tag number of an attribute use class to access the CDM for its conceptual type, size, and number of decimal digits.

22.2.1. Inputs

1. MODEL-NO PIC S9(4) COMP

MODEL-NO contains the number of the integrated model.

2. TAG-NO PIC S9(4) COMP

TAG-NO contains the tag number of the attribute use class.

3. ERROR-FILE PIC X(30)

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

22.2.2 CDM Requirements

Attribute Class - ATTRIBUTE_CLASS
Attribute Use Class - ATTRIBUTE_USE_CL
Owned Attribute Class - OWNED_ATTRIBUTE
Data Type - USER_DEFINED_DATA_TYPE

22.2.3 Internal Requirements

None

22.2.4 Processing

1. Access the CDM entity classes ATTRIBUTE_CLASS, OWNED_ATTRIBUTE and ATTRIBUTE_USE_CL with a tag number and retrieve the domain number (DOMAIN_NO) and the entity class number (EC_NO). Using the domain number and data type indicator (DATA_TYPE_IND) equal to "STD" access the CDM entity class USER_DEFINED_DATA_TYPE and retrieve the type (TYPE_ID), size (SIZE) and number of decimal digits (NO_OF_DEC) for the tag number.

22.2.5 Output

1. EC-NO PIC S9(4) COMP

EC-NO contains the entry class number.

2. Conceptual Schema Format.

```
01    CS-TYPE      PIC X.
01    CS-SIZE      PIC 9(3).
01    CS-ND        PIC 9(2).
```

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

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

4. ERROR-STATUS

ERROR-STATUS is set to "1" whenever an error occurs.

5. RET-STATUS PIC X(5)

RET-STATUS will contain a return status value as described in the ERRCDM copy member.

22.3 CDCWF Combine Generator Work Files

This routine will combine the two work files used in generating the query processor into one file. It will append the second work file (procedure division code) onto the end of the first work file.

22.3.1 Inputs

1. Work File Names

```
01  WORK-FILE-NAME1      PIC X(30).
01  WORK-FILE-NAME2      PIC X(30).
```

2. Generator Host

```
01  HOST                 PIC X(3).
```

22.3.2 CDM Requirements

None

22.3.3 Internal Requirements

None

22.3.4 Processing

1. Read work file two (WORK-FILE-NAME2) and write each record to the end of work file one (WORK-FILE-NAME1).

2. Delete work file two

```
Call "DELFIL" with the following:
     HOST
     WORK-FILE-NAME2
```

22.3.5 Outputs

1. An updated WORK-FILE-NAME1 containing the contents of WORK-FILE-NAME2.

22.4 CDIC Generate Internal/Conceptual Transformation

This routine generates COBOL source code according to the ANSI X3.23-1974 standard into a SQL-based request processor for the internal to conceptual transformation.

This code generates the interface and calls to user defined complex mapping algorithms, if defined, for the particular data field(s)/tag(s) or record type/tag(s) combinations.

If no complex mapping algorithm is defined, simple moves are generated from the internal data fields or records to the conceptual attributes. The conceptual null flags are set whenever the database or user defined null values are encountered.

22.4.1 Inputs

1. WORK-FILE PIC X(30)

   WORK-FILE contains the name of the file into which the COBOL statements will be generated.

2. SUBTRANS-ID PIC 999

   SUBTRANS-ID contains the subtransaction identifier.

3. IS-ACTION-LIST in ISAL copy member of IISSCLIB

   IS-ACTION-LIST contains the internal schema representation of the data items to be retrieved.

4. CMA-TABLE in copy member of IISSCLIB

   CMA-TABLE contains complex mapping algorithm parameter information.

5. NUMERIC-NULL PIC X(30)

   NUMERIC-NULL contains the user specified numeric null value or null to indicate that the database null value is to be used.

6. CHAR-NULL PIC X(30)

   CHAR-NULL contains the user specified character null value or null to indicate that the database null value is to be used.

7. CS-QUALIFY-LIST in CSQUAL copy member of IISSCLIB

   CS-QUALIFY-LIST contains the conceptual representation of the where clause.

8. IS-QUALIFY-LIST in ISQUAL copy of IISSCLIB

   IS-QUALIFY-LIST contains the internal representation of the where clause.

22.4.2 CDM Requirements

None

22.4.3 Internal Requirements

None

Macro Generation

Macros are code templates with optional substitutable parameters which allow generated code to be more independent of the generating programs. All macros are to be generated through calls to CDMACR. This routine requires the following parameters:

```
Input
 FILE-NAME        PIC X(30)      included in MACDAT copy member
 LIBRARY-NAME     PIC X(30)      included in MACDAT copy member
 MACRO-NAME       PIC X(8)       included in MACDAT copy member
 SUBSTITUTION-LIST               included in SBSTLST copy member

 Output
  RET-STATUS           PIC X(5)
```

FILE-NAME contains the name of the file to which code is generated. This file must be closed prior to the CDMACR call. Upon return to CDIC, FILE-NAME must be reopened for EXTEND to allow code to be generated at the end of the file.

LIBRARY-NAME contains the name of the library containing CDIC macros. All CDIC macros have a library name of SQL.

MACRO-NAME contains the name of the macro to be generated, e.g. CDIC01.

SUBSTITUTION-LIST is described by the following structure:

```
01   SUBSTITUTION-LIST.
   03   SL-USED       PIC 99.
   03   SL-MAX        PIC 99.
   03   SL-ROW-SIZE   PIC 99.
   03   SL-ENTRY OCCURS 8 TIMES
        INDEXED BY SL-INDEX.
     05   SL-PARAMETER PIC X(30).
     05   SL-SUBST-VAL PIC X(30).
```

SUBSTITUTION-LIST is populated by setting SL-USED to the number of parameter values the macro requires. SL-PARAMETER (index) contains the macro parameter to be substituted, e.g. P1. SL-SUBST-VAL (index) contains the corresponding substitution value, e.g. IS-DFNO.

22.4.4 Processing

1. Open EXTEND WORK-FILE.

   Move KES-SUCCESSFUL to RET-STATUS.

2. If IS-ACTION equal D or M, continue processing at step 6.

3. Transform each non complex IS-ACTION entry.

   For each used IS-ACTION entry with IS-ALG-PTR equal zero, perform the following steps. When all non-complex entries have been transformed, go to step 4.

   3.1 Generate a test to determine whether a database null value was returned for the current data field by substituting the current IS-DFNO for parameter P1 and IS-RTNO for P2 in macro CDIC01.

   3.2 If the data field's data type is character and the user specified character null value is not the database null value (IS-DATA-TYPE equal C and CHAR-NULL not equal NULL), generate the following check by substituting the current IS-DFNO for parameter P1, IS-RTNO for P2 and CHAR-NULL for P3 in macro CDIC02.

   3.3 If the data field's data type is not character and the user specified numeric null value is not the database null value (IS-DATA-TYPE not equal C and NUMERIC-NULL not equal NULL), generate the following check by substituting the current IS-DFNO for parameter P1, IS-RTNO for P2 and NUMERIC-NULL for P3 in macro CDIC02.

   3.4 Generate the local null flag test by substituting the current IS-RFT-PTR for parameter P1, IS-DFNO for P2 and IS-RTNO for P3 in macro CDIC03.

4. Transform each complex mapping algorithm for the subtransaction.

   Scan the CMA-ALG-ENTRY entries looking for any entries which have CMA-SUBTRANSACTION equal SUBTRANS-ID. For each such entry, perform the following steps. When all entries have been transformed, go to step 5.

   4.1 Determine the complex mapping type.

   Scan the current CMA-PARM-ENTRY entries looking for a non-zero CMA-RT-NO which has a corresponding CMA-DF-NO not equal zero. If found, this is a datafield(s) to tag(s) complex mapping. Perform steps 4.2 through 4.13.

   If the non-zero CMA-RT-NO has a corresponding CMA-DF-NO equal zero, a record type to tag(s) mapping exists. Perform steps 4.14 through 4.25.

4.2 For each non-zero CMA-DF-NO entry in the current CMA-ALG-ENTRY, generate a database null check by substituting the value of the non-zero CMA-DF-NO for parameter P1 and the corresponding CMA-RT-NO for P2 in macro CDIC01.

4.3 For each non-zero CMA-DF-NO entry in the current CMA-ALG-ENTRY, determine whether a user specified null value check must be generated.

If the non-zero CMA-DF-NO's CMA-DF-TYPE equals C and CHAR-NULL is not equal NULL, generate a user null test by substituting the CMA-DF-NO for parameter P1, CMA-RT-NO for P2 and CHAR-NULL for P3 in macro CDIC02.

If the non-zero CMA-DF-NO's CMA-DF-TYPE does not equal C and NUMERIC-NULL is not equal NULL, generate a user null test by substituting the CMA-DF-NO for parameter P1, CMA-RT-NO for P2 and NUMERIC-NULL for P3 in macro CDIC02.

4.4 Generate the local null flag test:

```
IF LOCAL-NULL-FLAG NOT = ZERO
```

Scan the IS-ACTION-LIST searching for all entries satisfying the following conditions:

```
IS-ALG-PTR = the current CMA-INDEX
```

and

```
IS-RFT-PTR not = zero.
```

For each entry found, generate the following 2 lines:

```
MOVE ZERO TO RES-isrft
MOVE 1 TO RES-NULL-isrft
```

where isrft is the value of the current IS-RFT-PTR.

After the preceeding moves have been generated for each qualifying entry, generate the following 2 lines:

```
MOVE ZERO TO LOCAL-NULL-FLAG
GO TO mod-inst.
```

where mod is the value of CMA-MOD-ID and inst is the value of CMA-MOD-INST.

4.5 For each non-zero CMA-DF-NO in the current CMA-ALG-ENTRY, generate a MOVE statement as follows:

MOVE D-dfno-rtno TO PARM-mod-inst-pno.

where dfno is the value of CMA-DF-NO, rtno is the value of CMA-RT-NO, mod is the value of CMA-MOD-ID, inst is the value of CMA-MOD-INST and pno is the value of CMA-PARM-NO.

4.6 Scan the current CMA-ALG-ENTRY searching for a CMA-PARM-ENTRY which has CMA-TAG-NO, CMA-RT-NO and CMA-DF-NO equal zero. For each such entry found, if any, generate one of the following two MOVE statements:

If CMA-PARM-TYPE equals C, generate:

MOVE "constval" TO PARM-mod-inst-pno.

Place the quote marks around the CMA-CONST-VAL to generate a character literal.

If CMA-PARM-TYPE does not equal C, generate:

MOVE constval TO PARM-mod-inst-pno.

where constval is the value of CMA-CONST-VAL, mod is the value of CMA-MOD-ID, inst is the value of CMA-MOD-INST and pno is the value of CMA-PARM-NO.

4.7 Generate the following complex mapping algorithm module call statement.

CALL "mod" USING

where mod is the CMA-MOD-ID value.

4.8 Generate the parameter list by generating, for each CMA-PARM-ENTRY in CMA-PARM-NO order, 1 line as follows:

PARM-mod-inst-pno

where mod is the value of the current CMA-MOD-ID, inst is the value of the current CMA-MOD-INST and pno is the value of the current CMA-PARM-NO.

4.9 Generate the status parameter and terminating period.

RET-STATUS.

4.10 Generate the status checking logic by substituting the current CMA-MOD-ID for parameter P1 in macro CDIC04.

4.11 Generate the moves into the results fields and null flags. For each non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY, generate the following 2 MOVE statements.

MOVE PARM-mod-inst-pno TO RES-isrft.

MOVE ZERO TO RES-NULL-isrft.

where mod is the value of the current CMA-MOD-ID, inst is the value of CMA-MOD-INST, pno is the current CMA-PARM-NO value and isrft is the value of the IS-RFT-PTR which is found by scanning the IS-ACTION-LIST searching for an entry in which

IS-ALG-PTR = current CMA-INDEX

and

IS-PARM-NO = current CMA-PARM-NO.

4.12 Generate a label according to the following format:

mod-inst.

where mod is the value of CMA-MOD-ID and inst is the value of CMA-MOD-INST.

4.13 Go to step 4.1 to process the next qualifying CMA-ALG-ENTRY, if any.

4.14 Generate database null tests for each datafield which makes up the record in this read type to tag(s) mapping.

Search the CMA-DF-ENTRY entries searching for all entries which have DF-MOD-PTR equal CMA-INDEX (at least 1 will be found).

For each entry found, perform the following steps.

4.14.1 Generate a database null test by substituting the current DF-DFNO for P1 and CMA-RT-NO for P2 in macro CDIC01.

4.14.2 If the CMA-RT-NO equals C and CHAR-NULL does not equal NULL, generate a user null test by substituting the current DF-DFNO for parameter P1, CMA-RT-NO for P2 and the CHAR-NULL value for P3 in macro CDIC02.

4.14.3 If the DF-TYPE does not equal C and NUMERIC-NULL does not equal NULL, generate a user null test by substituting the current DF-DFNO for parameter P1, CMA-RT-NO for P2 and the NUMERIC-NULL value for P3 in macro CDIC02.

4.15 Generate the local null flag test. Generate the following:

```
IF LOCAL-NULL-FLAG NOT = ZERO
```

Scan the IS-ACTION-LIST searching for all entries which have IS-ALG-PTR matching the current CMA-INDEX and IS-RFT-PTR not equal zero.

For each IS entry, generate the following two lines:

```
MOVE ZERO TO RES-isrft
MOVE 1 TO RES-NULL-isrft
```

where isrft is the current IS-RFT-PTR value.

Generate the following 2 lines:

```
MOVE ZERO TO LOCAL-NULL-FLAG
GO TO mod-inst.
```

where mod is the current CMA-MOD-ID value and inst is the CMA-MOD-INST value.

4.16 Scan the CMA-DF-ENTRY entries searching for all entries which have DF-MOD-PTR equal CMA-INDEX. For each entry found, generate the following MOVE statement:

```
MOVE D-dfno-rtno TO D-dfno
OF T-rtno.
```

where dfno is the current DF-DFNO value and rtno is the CMA-RT-NO value.

4.17 Generate the move of the record to the parameter.

Scan the current CMA-ALG-ENTRY searching for the non-zero CMA-RT-NO. When found, generate the following line:

```
MOVE T-rtno TO PARM-mod-inst-pno.
```

where rtno is the current CMA-RT-NO value, mod is the CMA-MOD-ID value, inst is the CMA-MOD-INST value and pno is the CMA-PARM-NO value.

4.18 Generate MOVE statements for complex mapping constants, if any.

Scan the current CMA-ALG-ENTRY searching for a CMA-PARM-ENTRY which has CMA-TAG-NO, CMA-RT-NO and CMA-DF-NO equal zero.

For each such entry found, generate one of the following two MOVE statements.

If the CMA-PARM-TYPE equals C, generate

```
MOVE "constval" TO PARM-mod-inst-pno.
```

Place the quote marks around the CMA-CONST-VAL to generate a character literal.

If the CMA-PARM-TYPE does not equal C, generate

```
MOVE constval TO PARM-mod-inst-pno.
```

where constval is the current CMA-CONST-VAL, mod is the current CMA-MOD-ID, inst is the current CMA-MOD-INST value and pno is the current CMA-PARM-NO value.

4.19 Generate the module call.

```
CALL "mod" USING
```

where mod is the current CMA-MOD-ID value.

4.20 Generate the parameter list by generating, for each CMA-PARM-ENTRY in CMA-PARM-NO order, the following line:

```
PARM-mod-inst-pno
```

Where mod is the current CMA-MOD-ID value, inst is the CMA-MOD-INST value and pno is the CMA-PARM-NO value.

4.21 Generate the status parameter and period.

```
RET-STATUS.
```

4.22 Generate the status checking logic by substituting the current CMA-MOD-ID for parameter P1 in macro CDIC04.

4.23 Generate the moves into the results fields and null flags. For each non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY, generate the following two MOVE statements:

```
MOVE PARM-mod-inst-pno TO RES-isrft.
MOVE ZERO TO RES-NULL-isrft.
```

where mod is the current CMA-MOD-ID value, inst is the CMA-MOD-INST value, pno is the CMA-PARM-NO value and isrft is the IS-RFT-PTR value found by scanning the IS-ACTION-LIST searching for an entry which has IS-ALG-PTR equal to the current CMA-INST and IS-PARM-NO equal to the current CMA-PARM-NO.

4.24 Generate the following label:

mod-inst.

where mod is the CMA-MOD-ID value and inst is the CMA-MOD-INST value.

4.25 Go to step 4.1 to process the next qualifying CMA-ALG-ENTRY, if any.

5. If IS-ACTION does not equal D or M, go to step 6.

5.1 Determine whether all IS-QUALIFY entries are internally evaluatable by scanning the IS-QUALIFY-LIST searching for a used entry with ISQ-EVAL-FLAG equal zero.

If no entries are found with ISQ-EVAL-FLAG equal zero, no qualify entries need be transformed. Go to step 6.

If at least 1 entry has ISQ-EVAL-FLAG equal zero, all ISQ entries which are not union discriminators must be transformed.

5.2 Scan the IS-QUALIFY-LIST. For each used ISQ entry which has ISQ-TYPE = 2 and ISQ-CSQ-PTR not equal zero, set ISQ-LEFT TO 1.

For each used ISQ entry which has ISQ-TYPE = 3 set ISQ-LEFT and ISQ-RIGHT TO 1.

5.3 Perform the following steps for each marked IS-QUALIFY entry. When all marked IS-QUALIFY entries have been processed, go to step 6.

5.4 If ISQ-TYPE equals 2, perform the following steps.

5.4.1 Determine whether the mapping is complex.

If ISQ-ALG-PTRL not equal zero, the mapping is complex. Go to step 5.4.3.

5.4.2 Transform the non-complex IS-QUALIFY entry.

5.4.2.1 Generate a test to determine whether a database null value was returned for the current IS-QUALIFY entry by substituting the current ISQ-DFNOL for parameter P1 and ISQ-RTNOL for P2 in macro CDIC01.

5.4.2.2 If the entry's data type is character and the user specified character null value is not the database null value (ISQ-TYPEL equal C and CHAR-NULL not equal NULL), generate the following check by substituting the current ISQ-DFNOL for parameter P1: ISQ-RTNOL for P2 and CHAR-NULL for P3 in macro CDIC02.

5.4.2.3 If the entry's data type is not character and the user specified numeric null value is not the database null value (ISQ-TYPEL not equal C and NUMERIC-NULL not equal NULL), generate the following check by substituting the current ISQ-DFNOL for parameter P1: ISQ-RTNOL for P2 and NUMERIC-NULL for P3 in macro CDIC02.

5.4.2.4 Generate the local null flag test by substituting the CSQ-AUCL pointed to by the current ISQ-CSQ-PTR as parameter P1, ISQ-DFNOL for P2 and ISQ-RTNOL for parameter P3 in macro CDIC05.

5.4.2.5 Scan the remaining marked IS-QUALIFY entries searching for ISQ-DFNOL/ISQ-RTNOL or ISQ-DFNOR/ISQ-RTNOR combinations which equal the current ISQ-DFNOL/ISQ-RTNOL combination and whose corresponding ISQ-ALG-PTR-L or ISQ-ALG-PTR-R equals zero. For each match found, set the corresponding ISQ-LEFT or ISQ-RIGHT to zero.

5.4.2.6 Set the current ISQ-LEFT to zero.

5.4.2.7 Go to step 5.3 to process the next marked ISQ entry.

5.4.3 A complex mapping exists for the current ISQ entry. Determine whether the mapping is a datafield(s) to tag or record type to tag mapping.

Scan the CMA-PARM-ENTRYs pointed to by the current ISQ-ALG-PTR-L looking for an entry which has a CMA-RT-NO not equal zero with a corresponding CMA-DF-NO equal zero. If found, the mapping is a record type to tag mapping. Go to step 5.4.3.15.

5.4.3.1 For each non-zero CMA-DF-NO entry pointed to by the current ISQ-ALG-PTR-L, generate a database null check by substituting the value of the non-zero CMA-DF-NO for parameter P1 and the corresponding CMA-RT-NO for P2 in macro CDIC01.

5.4.3.2 For each non-zero CMA-DF-NO entry pointed to by ISQ-ALG-PTR-L, determine whether a user specified null value check must be generated.

If the non-zero CMA-DF-NO's CMA-DF-TYPE equals C and CHAR-NULL is not equal NULL, generate a user null test by substituting the CMA-DF-NO for P1, CMA-RT-NO for P2 and CHAR-NULL for P3 in macro CDIC02.

If the non-zero CMA-DF-NO's CMA-DF-TYPE does not equal C and NUMERIC-NULL is not equal NULL, generate a user null test by substituting the CMA-DF-NO for parameter P1, CMA-RT-NO for P2 and NUMERIC-NULL for P3 in macro CDIC02.

5.4.3.3 Generate the local null flag test by substituting the only non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY for P1, the current CMA-MOD-ID for P2 and the current CMA-MOD-INST for P3 in macro CDIC06.

5.4.3.4 For each non-zero CMA-DF-NO in the current CMA-ALG-ENTRY, generate the following statement:

MOVE D-dfno-rtno TO PARM-mod-inst-pno.

Where dfno is the value of CMA-DF-NO, rtno is the value of CMA-RT-NO, mod is the value of CMA-MOD-ID and pno is the value of CMA-PARM-NO.

5.4.3.5 Scan the current CMA-ALG-ENTRY searching for a CMA-PARM-ENTRY which has CMA-TAG-NO, CMA-RT-NO and CMA-DF-NO equal zero. For each such entry found, generate one of the following MOVE statements:

If CMA-PARM-TYPE equals C, generate:

MOVE "constval" TO PARM-mod-inst-pno.

Place the quote marks around the CMA-CONST-VAL to generate a literal character.

If CMA-PARM-TYPE does not equal C, generate:

MOVE constval TO PARM-mod-inst-pno.

Where constval is the value of CMA-CONST-VAL, mod is the value of CMA-MOD-ID, inst is the value of CMA-MOD-INST and pno is the value of CMA-PARM-NO.

5.4.3.6 Generate the following CMA call statement:

CALL "mod" USING

where mod is the CMA-MOD-ID value.

5.4.3.7 Generate the parameter list by generating, for each CMA-PARM-ENTRY in CMA-PARM-NO order, 1 line as follows:

PARM-mod-inst-pno

where mod is the value of the current CMA-MOD-ID, inst is the value of the current CMA-MOD-INST and pno is the value of the current CMA-PARM-NO.

5.4.3.8 Generate the status parameter and terminating period.

RET-STATUS.

5.4.3.9 Generate the status checking logic by substituting the current CMA-MOD-ID for parameter P1 in macro CDIC04.

5.4.3.10 Generate the moves into the conceptual field and flag for the non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY.

```
MOVE PARM-mod-inst-pno TO TAG-tno.
MOVE ZERO TO TAG-NULL-tno.
```

where mod is the value of the current CMA-MOD-ID, inst is the value of CMA-MOD-INST, pno is the current CMA-PARM-NO value and tno is the only non-zero CMA-TAG-NO value in the current CMA-ALG-ENTRY.

5.4.3.11 Generate a label.

mod-inst.

where mod is the current CMA-MOD-ID value and inst is the current CMA-MOD-INST value.

5.4.3.12 Scan the remaining marked IS-QUALIFY entries searching all ISQ-ALG-PTR-Ls and ISQ-ALG-PTR-Rs matching the current ISQ-ALG-PTR-L. For each match found, set the corresponding ISQ-LEFT or ISQ-RIGHT to zero.

5.4.3.13 Set the current ISQ-LEFT to zero.

5.4.3.14 Go to step 5.3 to process the next marked ISQ-entry.

5.4.3.15 Transform the record type to tag mapping.

Generate database null tests for each data field of the record type. Search the CMA-DF-ENTRY entries searching for all entries which have DF-MOD-PTR equal ISQ-ALG-PTRL (ISQ-index) and perform the following steps:

5.4.3.15.1 Generate a native null test by substituting the current DF-DFNO for P1 and the current CMA-RT-NO for P2 in macro CDIC01.

5.4.3.15.2 If the current DF-TYPE equals C and CHAR-NULL does not equal NULL, generate a user null test by substituting the current DF-DFNO for P1, CMA-RT-NO for P2 and the CHAR-NULL value for P3 in macro CDIC02.

5.4.3.15.3 If the current DF-TYPE does not equal C and NUMERIC-NULL does not equal NULL, generate a user null test by substituting the current DF-DFNO for P1, CMA-RT-NO for P2 and the NUMERIC-NULL value for P3 in macro CDIC02.

5.4.3.16 Generate the local null flag test by substituting the only non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY for P1, CMA-MOD-ID for P2 and CMA-MOD-INST for P3 in macro CDIC06.

5.4.3.17 Generate moves of the data fields to the record type structure for each used CMA-DF-ENTRY.

MOVE D-dfno-rtno TO D-dfno OF T-rtno.

where dfno is the current DF-DFNO value and rtno is the current CMA-RT-NO value.

5.4.3.18 Generate the move of the record to the parameter.

Scan the current CMA-ALG-ENTRY searching for the non-zero CMA-RT-NO. When found, generate the following line:

MOVE T-rtno TO PARM-mod-inst-pno.

where rtno is the current CMA-RT-NO value, mod is the CMA-MOD-ID value, inst is the CMA-MOD-INST value and pno is the CMA-PARM-NO value.

5.4.3.19 Generate MOVE statements for complex mapping constants, if any.

Scan the current CMA-ALG-ENTRY searching for a CMA-PARM-ENTRY which has CMA-TAG-NO, CMA-RT-NO and CMA-DF-NO equal zero.

For each such entry found, generate one of the following two MOVE statements.

If the CMA-PARM-TYPE equals C, generate

MOVE "constval" TO PARM-mod-inst-pno.

Place the quote marks around the CMA-CONST-VAL to generate a literal character.

If the CMA-PARM-TYPE does not equal C, generate

MOVE constval TO PARM-mod-inst-pno.

where constval is the current CMA-CONST-VAL, mod is the current CMA-MOD-ID, inst is the current CMA-MOD-INST value and pno is the current CMA-PARM-NO value.

5.4.3.20 Generate the module call.

```
CALL "mod" USING
```

where mod is the current CMA-MOD-ID value.

5.4.3.21 Generate the parameter list by generating, for each CMA-PARM-ENTRY in CMA-PARM-NO order, the following line:

```
PARM-mod-inst-pno
```

where mod is the current CMA-MOD-ID value, inst is the CMA-MOD-INST value and pno is the CMA-PARM-NO value.

5.4.3.22 Generate the status parameter and period.

```
RET-STATUS.
```

5.4.3.23 Generate the status checking logic by substituting the current CMA-MOD-ID for parameter P1 in macro CDIC04.

5.4.3.24 Generate the moves into the conceptual field and flag for the non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY.

```
MOVE PARM-mod-inst-pno TO
TAG-tno.
MOVE ZERO TO TAG-NULL-tno.
```

where mod is the CMA-MOD-ID value, inst is the CMA-MOD-INST value, pno is the current CMA-PARM-NO value and tno is the only non-zero CMA-TAG-NO value in the current CMA-ALG-ENTRY.

5.4.3.25 Generate a label.

```
mod-inst.
```

where mod is the CMA-MOD-ID value and inst is the CMA-MOD-INST value.

5.4.3.26 Scan the remaining marked IS-QUALIFY entries searching for all ISQ-ALG-PTR-Ls and ISQ-ALG-PTR-Rs matching the current ISQ-ALG-PTR-L. For each match found, set the corresponding ISQ-LEFT or ISQ-RIGHT to zero.

5.4.3.27 Set the current ISQ-LEFT to zero.

5.4.3.28 Go to step 5.3 to process the next marked ISQ entry.

5.5 If ISQ-TYPE equals 3, perform the following steps:

5.5.1 Transform the left side.

5.5.1.1 Determine whether the left side mapping is complex.

If ISQ-ALG-PTR-L not equal zero, the left side mapping is complex. Go to step 5.5.1.3.

5.5.1.2 Transform the non-complex IS-QUALIFY left side entry.

5.5.1.2.1 Generate a test to determine whether a database null value was returned for the current IS-QUALIFY left entry by substituting the current ISQ-DFNOL for parameter P1 and ISQ-RTNOL for P2 in macro CDIC01.

5.5.1.2.2 If the left entry's type is character and the user specified character null value is not the database null value (ISQ-TYPEL equal C and CHAR-NULL not

equal NULL), generate the following check by substituting the current ISQ-DFNOL for parameter P1, ISQ-RTNOL for P2 and CHAR-NULL for P3 in macro CDIC02.

5.5.1.2.3 If the left entry's data type is not character and the user specified numeric null value is not the database null value (ISQ-TYPEL not equal C and NUMERIC-NULL not equal NULL), generate the following check by substituting the current ISQ-DFNOL for parameter P1, ISQ-RTNOL for P2 and NUMERIC-NULL for P3 in macro CDIC02.

5.5.1.2.4 Generate the local null flag test by substituting the CSQ-AUCL pointed to by the current ISQ-CSQ-PTR as parameter P1, ISQ-DFNOL for P2 and ISQ-RTNOL for parameter P3 in macro CDIC05.

5.5.1.2.5 Scan the remaining marked IS-QUALIFY entries searching for ISQ-DFNOL/ISQ-RTNOL or ISQ-DFNOR/ISQ-RTNOR combinations which equal the current ISQ-DFNOL/ISQ-RTNOL combination and whose corresponding ISQ-ALG-PTR-L or

ISQ-ALG-PTR-R equals zero. For each match found, set the corresponding ISQ-LEFT or ISQ-RIGHT to zero.

5.5.1.2.6 Set the current ISQ-LEFT to zero.

5.5.1.2.7 Go to step 5.5.2 to transform the right side if ISQ-RIGHT = 1.

5.5.1.3 A complex mapping exists for the current ISQ left entry. Determine whether the mapping is a data field(s) to tag or record type to tag mapping.

Scan the CMA-PARM-ENTRYs pointed to by the current ISQ-ALG-PTR-L looking for an entry which has a CMA-RT-NO not equal zero with a corresonding CMA-DF-NO equal zero. If found, the mapping is a record type to tag mapping. Go to step 5.5.1.3.14.

5.5.1.3.1 For each non-zero CMA-DF-NO entry pointed to by the current ISQ-ALG-PTR-L, generate a database null check by substituting the value of the non-zero CMA-DF-NO for parameter P1 and the corresponding CMA-RT-NO for P2 in macro CDIC01.

5.5.1.3.2 For each non-zero CMA-DF-NO entry pointed to by ISQ-ALG-PTRL, determine whether a user specified null value check must be generated.

If the non-zero CMA-DF-NO's CMA-DF-TYPE equals C and CHAR-NULL is not equal NULL, generate a user null test by substituting the CMA-DF-NO for P1, CMA-RT-NO for P2 and CHAR-NULL for P3 in macro CDIC02.

If the non-zero CMA-DF-NO's CMA-DF-TYPE does not equal C and NUMERIC-NULL is not equal NULL, generate a user null test by substituting the CMA-DF-NO for parameter P1, CMA-RT-NO for P2 and NUMERIC-NULL for P3 in macro CDIC02.

5.5.1.3.3 Generate the local null flag test by substituting the only non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY for P1, the current CMA-MOD-ID for P2 and the current CMA-MOD-INST for P3 in macro CDIC06.

5.5.1.3.4 For each non-zero CMA-DF-NO in the current CMA-ALG-ENTRY, generate the following statement:

MOVE D-dfno-rtno
TO
PARM-mod-inst-pno.

where dfno is the value of CMA-DF-NO, rtno is the value of CMA-RT-NO, mod is the value of CMA-MOD-ID and pno is the value of CMA-PARM-NO.

5.5.1.3.5 Scan the current CMA-ALG-ENTRY searching for a CMA-PARM-ENTRY which has CMA-TAG-NO, CMA-RT-NO and CMA-DF-NO equal zero. For each such entry found, generate one of the following MOVE statements.

If CMA-PARM-TYPE equals C, generate:

MOVE "constval" TO PARM-mod-inst-pno.

Place the quote marks around the CMA-CONST-VAL to generate a character literal.

If CMA-PARM-TYPE does not equal C, generate:

MOVE constval TO PARM-mod-inst-pno.

where constval is the value of CMA-CONST-VAL, mod is the value of CMA-MOD-ID, inst is the value of CMA-MOD-INST and pno is the value of CMA-PARM-NO.

5.5.1.3.6 Generate the following CMA call statement:

CALL "mod" USING

where mod is the CMA-MOD-ID value.

5.5.1.3.7 Generate the parameter list by generating, for each CMA-PARM-ENTRY in CMA-PARM-NO order, 1 line as follows:

PARM-mod-inst-pno

where mod is the value of the current CMA-MOD-ID, inst is the value of the current CMA-MOD-INST and pno is the value of the current CMA-PARM-NO.

5.5.1.3.8 Generate the status parameter and terminating period.

RET-STATUS.

5.5.1.3.9 Generate the status checking logic by substituting the current CMA-MOD-ID for parameter P1 in macro CDIC04.

5.5.1.3.10 Generate the moves into the conceptual field and flag for only non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY.

MOVE
PARM-mod-inst-pno
TO TAG-tno.
MOVE ZERO TO
TAG-NULL-tno.

where mod is the value of the current CMA-MOD-ID, inst

is the value of CMA-MOD-INST, pno is the current CMA-PARM-NO value and tno is the only non-zero CMA-TAG-NO value in the current CMA-ALG-ENTRY.

5.5.1.3.11 Generate a label.

mod-inst.

where mod is the current CMA-MOD-ID value and inst is the current CMA-MOD-INST value.

5.5.1.3.12 Scan the remaining marked IS-QUALIFY entries searching for all ISG-ALG-PTR-Ls and ISQ-ALG-PTR-Rs matching the current ISQ-ALG-PTR-L. For each match found, set the corresponding ISQ-LEFT or ISQ-RIGHT to zero.

5.5.1.3.13 Set the current ISQ-LEFT to zero and go to step 5.5.2

5.5.1.3.14 Transform the record type to tag mapping.

Generate database null tests for each data field of the record type. Search the CMA-DF-ENTRY entries searching for all entries which have DF-MOD-PTR equal ISQ-ALG-PTRL (ISQ-index) and perform the following steps:

- Generate a native null test by substituting the current DF-DFNO for P1 and the current CMA-RT-NO for P2 in macro CDIC01.
- If the current DF-TYPE equals C and CHAR-NULL does not equal NULL, generate a user null test by substituting the current DF-DFNO for P1, CMA-RT-NO for P2 and the CHAR-NULL value for P3 in macro CDIC02.
- If the current DF-TYPE does not equal C and NUMERIC-NULL does not equal NULL, generate a user null test by substituting the current DF-DFNO for P1, CMA-RT-NO for P2 and the NUMERIC-NULL value for P3 in macro CDIC02.

5.5.1.3.15 Generate the local null flag test by substituting the only non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY for P1, CMA-MOD-ID for P2 and CMA-MOD-INST for P3 in macro CDIC06.

5.5.1.3.16 Generate moves of the data fields to the record type structure for each used CMA-DF-ENTRY.

MOVE D-dfno-rtno
TO D-dfno OF
T-rtno.

where dfno is the current DF-DFNO value and rtno is the current CMA-RT-NO value.

5.5.1.3.17 Generate the move of the record to the parameter.

Scan the current CMA-ALG-ENTRY searching for the non-zero CMA-RT-NO. When found, generate the following line:

MOVE T-rtno TO
PARM-mod-inst-pno.

where rtno is the current CMA-RT-NO value, mod is the CMA-MOD-ID value, inst is the CMA-MOD-INST value and pno is the CMA-PARM-NO value.

5.5.1.3.18 Generate MOVE statements for complex mapping constants, if any.

Scan the current CMA-ALG-ENTRY searching for a CMA-PARM-ENTRY which has CMA-TAG-NO, CMA-RT-NO and CMA-DF-NO equal zero.

For each such entry found, generate one of the following two MOVE statements.

If the CMA-PARM-TYPE equals C, generate

MOVE "constval" TO PARM-mod-inst-pno.

Place the quote marks around the CMA-CONST-VAL to generate a character literal.

If the CMA-PARM-TYPE does not equal C, generate

MOVE constval TO PARM-mod-inst-pno.

where constval is the current CMA-CONST-VAL, mod is the current CMA-MOD-ID, inst is the current CMA-MOD-INST value and pno is the current CMA-PARM-NO value.

5.5.1.3.19 Generate the module call.

CALL "mod" USING

where mod is the current CMA-MOD-ID value.

5.5.1.3.20 Generate the parameter list by generating, for each CMA-PARM-ENTRY in CMA-PARM-NO order, the following line:

PARM-mod-inst-pno

where mod is the current CMA-MOD-ID value, inst is the CMA-MOD-INST value and pno is the CMA-PARM-NO value.

5.5.1.3.21 Generate the status parameter and period.

RET-STATUS.

5.5.1.3.22 Generate the status checking logic by substituting the current CMA-MOD-ID for parameter P1 in macro CDIC04.

5.5.1.3.23 Generate the moves into the conceptual field and flag for the non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY.

MOVE
PARM-mod-inst-pno
TO TAG-tno.
MOVE ZERO TO
TAG-NULL-tno.

where mod is the CMA-MOD-ID value, inst is the CMA-MOD-INST value, pno is the current CMA-PARM-NO value and tno is the only non-zero CMA-TAG-NO value in the current CMA-ALG-ENTRY.

5.5.1.3.24 Generate a label.

mod-inst.

where mod is the CMA-MOD-ID value and inst is the CMA-MOD-INST value.

5.5.1.3.25 Scan the remaining marked IS-QUALIFY entries searching for all ISQ-ALG-PTR-Ls and ISQ-ALG-PTR-Rs matching the

current ISQ-ALG-PTR-L. For each match found, set the corresponding ISQ-LEFT or ISQ-RIGHT to zero.

5.5.1.3.26 Set the current ISQ-LEFT to zero.

5.5.2 Transform the right side, if marked.

If the current ISQ-RIGHT equals zero, go to step 5.3 to process the next marked ISQ entry.

5.5.2.1 Determine whether the right side mapping is complex.

If ISQ-ALG-PTR-R not equal zero, the mapping is complex. Go to step 5.5.2.3.

5.5.2.2 Transform the non-complex IS-QUALIFY right side entry.

5.5.2.2.1 Generate a test to determine whether a database null value returned for the current IS-QUALIFY entry by substituting the current ISQ-DFNOR for parameter P1 and ISQ-RTNOR for P2 in macro CDIC01.

5.5.2.2.2 If the entry's data type is character and the user specified character null value is not the database null value (ISQ-TYPER equal C and CHAR-NULL not equal NULL), generate the following check by substituting the current ISQ-DFNOR for parameter P1, ISQ-RTNOR for P2

and CHAR-NULL for P3 in macro CDIC02.

5.5.2.2.3 If the entry's data type is not character and the user specified numeric null value is not the database null value (ISQ-TYPER not equal C and NUMERIC-NULL not equal NULL), generate the following check by substituting the current ISQ-DFNOR for parameter P1, ISQ-RTNOR for P2 and NUMERIC-NULL for P3 in macro CDIC02.

5.5.2.2.4 Generate the local null flag test by substituting the CSQ-AUCR pointed to by the current ISQ-CSQ-PTR as parameter P1, ISQ-DFNOR for P2 and ISQ-RTNOR for parameter P3 in macro CDIC05.

5.5.2.2.5 Scan the remaining marked IS-QUALIFY entries searching for ISQ-DFNOL/ISQ-RTNOL or ISQ-DFNOR/ISQ-DFNOR combinations which equal the current ISQ-DFNOR/ISQ-RTNOR combination and whose corresponding ISQ-ALG-PTR-L or ISQ-ALG-PTR-R equals zero. For each match found, set the corresponding ISQ-LEFT or ISQ-RIGHT to zero.

5.5.2.2.6 Set the current ISQ-RIGHT to zero.

5.5.2.2.7 Go to step 5.3 to process the next marked ISQ entry.

5.5.2.3 A complex mapping exists for the current ISQ right entry. Determine whether the mapping is a data field(s) to tag or record type to tag mapping.

Scan the CMA-PARM-ENTRY(s) pointed to by the current ISQ-ALG-PTR-R looking for an entry which has a non-zero CMA-RT-NO with a corresponding CMA-DF-NO equal zero. If found, the mapping is a record type to tag mapping. Go to step 5.5.2.3.15.

5.5.2.3.1 For each non-zero CMA-DF-NO entry pointed to by the current ISQ-ALG-PTR-R, generate a database null check by substituting the value of the non-zero CMA-DF-NO for parameter P1 and the corresponding CMA-RT-NO for P2 in macro CDIC01.

5.5.2.3.2 For each non-zero CMA-DF-NO entry pointed to by ISQ-ALG-PTR-R, determine whether a user specified null value check must be generated.

If the non-zero CMA-DF-NO's CMA-DF-TYPE equals C and CHAR-NULL is not equal NULL, generate a user null test by substituting the CMA-DF-NO for P1,

CMA-RT-NO for P2 and CHAR-NULL for P3 in macro CDIC02.

If the non-zero CMA-DF-NO's CMA-DF-TYPE does not equal C and NUMERIC NULL is not equal NULL, generate a user null test by submitting the CMA-DF-NO for parameter P1, CMA-RT-NO for P2 and NUMERIC-NULL for P3 in macro CDIC02.

5.5.2.3.3 Generate the local null flag test by substituting the only non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY for P1, the current CMA-MOD-ID for P2 and the current CMA-MOD-INST for P3 in macro CDIC06.

5.5.2.3.4 For each non-zero CMA-DF-NO in the current CMA-ALG-ENTRY, generate the following statement.

MOVE D-dfno-rtno
TO
PARM-mod-inst-pno.

where dfno is the value of CMA-DF-NO, rtno is the value of CMA-RT-NO, mod is the value of CMA-MOD-ID and pno is the value of CMA-PARM-NO.

5.5.2.3.5 Scan the current CMA-ALG-ENTRY searching for a CMA-PARM-ENTRY which has CMA-TAG-NO, CMA-RT-NO and CMA-DF-NO equal zero. For each such entry found, generate one of the following MOVE statements.

If CMA-PARM-TYPE equals C, generate:

MOVE "constval" TO PARM-mod-inst-pno.

Place the quote marks around the CMA-CONST-VAL to generate a character literal.

IF CMA-PARM-TYPE does not equal C, generate:

MOVE constval TO PARM-mod-inst-pno.

where constval is the value of CMA-CONST-VAL, mod is the value of CMA-MOD-ID, inst is the value of CMA-MOD-INST and pno is the value of CMA-PARM-NO.

5.5.2.3.6 Generate the following CMA call statement.

CALL "mod" USING

where mod is the CMA-MOD-ID value.

5.5.2.3.7 Generate the parameter list by generating, for each

CMA-PARM-ENTRY in CMA-PARM-NO order, 1 line as follows.

PARM-mod-inst-pno

where mod is the value of the current CMA-MOD-ID, inst is the value of the current CMA-MOD-INST and pno is the value of the current CMA-PARM-NO.

5.5.2.3.8 Generate the status parameter and terminating period.

RET-STATUS.

5.5.2.3.9 Generate the status checking logic by substituting the current CMA-MOD-ID for parameter P1 in macro CDIC04.

5.5.2.3.10 Generate the moves into the conceptual field and flag for the only non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY.

MOVE
PARM-mod-inst-pno
TO TAG-tno.
MOVE ZERO TO
TAG-NULL-tno.

where mod is the value of the current CMA-MOD-ID, inst is the value of CMA-MOD-INST, pno is the current CMA-PARM-NO value tno is the only non-zero

CMA-TAG-NO value in the current CMA-ALG-ENTRY.

5.5.2.3.11 Generate a label.

mod-inst.

where mod is the current CMA-MOD-ID value and inst is the current CMA-MOD-INST value.

5.5.2.3.12 Scan the remaining IS-QUALIFY entries searching for all ISQ-ALG-PTR-Ls and ISQ-ALG-PTR-Rs matching the current ISQ-ALG-PTR-R. For each match found, set the corresponding ISQ-LEFT or ISQ-RIGHT TO ZERO.

5.5.2.3.13 Set the current ISQ-RIGHT to zero.

5.5.2.3.14 Go to step 5.3 to process the next marked ISQ entry.

5.5.2.3.15 Transform the record type to tag mapping.

Generate database null tests for each data field of the record type. For each used CMA-DF-ENTRY, perform the following steps:

. Generate a native null test by substituting the current DF-DFNO for P1 and the current CMA-RT-NO for P2 in macro CDIC01.

. If the current DF-TYPE equals C and CHAR-NULL does not equal NULL, generate a

user null test by substituting the current DF-DFNO for P1, CMA-RT-NO for P2 and the CHAR-NULL value for P3 in macro CDIC02.

. If the current DF-TYPE does not equal C and NUMERIC-NULL does not equal NULL, generate a user null test by substituting the current DF-DFNO for P1, CMA-RT-NO for P2 and the NUMERIC-NULL value for P3 in macro CDIC02.

5.5.2.3.16 Generate the local null flag test by substituting the only non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY for P1, CMA-MOD-ID for P2 and CMA-MOD-INST for P3 in macro CDIC06.

5.5.2.3.17 Generate moves of the data fields to the record type structure for each used CMA-DF-ENTRY.

MOVE D-dfno-rtno TO D-dfno OF T-rtno.

where dfno is the current DF-DFNO value and rtno is the current CMA-RT-NO value.

5.5.2.3.18 Generate the move of the record to the parameter.

Scan the current CMA-ALG-ENTRY searching for the non-zero CMA-RT-NO. When found, generate the following line:

MOVE T-rtno TO
PARM-mod-inst-pno.

where rtno is the current CMA-RT-NO value, mod is the CMA-MOD-ID value, inst is the CMA-MOD-INST value and pno is the CMA-PARM-NO value.

5.5.2..3.19 Generate MOVE statements for complex mapping constants, if any.

Scan the current CMA-ALG-ENTRY searching for a CMA-PARM-ENTRY which has CMA-TAG-NO, CMA-RT-NO and CMA-DF-NO equal zero.

For each such entry found, generate one of the following two MOVE statements.

If the CMA-PARM-TYPE equals C, generate

MOVE "constval" TO
PARM-mod-inst-pno.

Place the quote marks around the CMA-CONST-VAL to generate a character literal.

If the CMA-PARM-TYPE does not equal C, generate

MOVE constval TO
PARM-mod-inst-pno.

where constval is the current CMA-CONST-VAL, mod is the current CMA-MOD-ID, inst

is the current CMA-MOD-INST value and pno is the current CMA-PARM-NO value.

5.5.2.3.20 Generate the module call.

CALL "mod" USING

where mod is the current CMA-MOD-ID value.

5.5.2.3.21 Generate the parameter list by generating, for each CMA-PARM-ENTRY in CMA-PARM-NO order, the following line:

PARM-mod-inst-pno

where mod is the current CMA-MOD-ID value, inst is the CMA-MOD-INST value and pno is the CMA-PARM-NO value.

5.5.2.3.22 Generate the status parameter and period.

RET-STATUS.

5.5.2.3.23 Generate the status checking logic by substituting the current CMA-MOD-ID for parameter P1 in macro CDIC04.

5.5.2.3.24 Generate the moves into the conceptual field and flag for the non-zero CMA-TAG-NO in the current CMA-ALG-ENTRY.

MOVE
PARM-mod-inst-pno
TO TAG-tno.
MOVE ZERO TO
TAG-NULL-tno.

where mod is the CMA-MOD-ID value, inst is the CMA-MOD-INST value, pno is the current CMA-PARM-NO value and tno is the only non-zero CMA-TAG-NO value in the current CMA-ALG-ENTRY.

5.5.2.3.25 Generate a label.

mod-inst.

where mod is the CMA-MOD-ID value and inst is the CMA-MOD-INST value.

5.5.2.3.26 Scan the remaining marked IS-QUALIFY entries searching for all ISQ-ALG-PTR-Ls and ISQ-ALG-PTR-Rs matching the current ISQ-ALG-PTR-L. For each match found, set the corresponding ISQ-LEFT or ISQ-RIGHT to zero.

5.5.2.3.27 Set the current ISQ-RIGHT to zero.

5.5.2.3.28 Go to step 4.3 to process the next marked ISQ entry.

6. Close WORK-FILE and terminate processing.

22.4.5 Outputs

1. RET-STATUS PIC X(5)

RET-STATUS contains the CDIC completion status. A value equal to KES-SUCCESSFUL is defined in the ERRCDM copy member indicates success.

CDIC MACROS

LIBRARY: SQL
MACRO: CDIC01

```
    IF INDP-P1-P2 = -1
       MOVE 1 TO LOCAL-NULL-FLAG.
```

CDIC MACROS

LIBRARY: SQL
MACRO: CDIC02

```
    IF D-P1-P2 = P3
       MOVE 1 TO LOCAL-NULL-FLAG.
```

CDIC MACROS

LIBRARY: SQL
MACRO: CDIC03

```
    IF LOCAL-NULL-FLAG = ZERO
       MOVE ZERO TO RES-NULL-P1

       MOVE D-P2-P3 TO RES-P1
    ELSE
       MOVE 1 TO RES-NULL-P1
       MOVE ZERO TO RES-P1
       MOVE ZERO TO LOCAL-NULL-FLAG.
```

CDIC MACROS

LIBRARY: SQL
MACRO: CDIC04

```
    IF RET-STATUS NOT = KES-SUCCESSFUL
       STRING "P1"
       " TRANSFORM PROGRAM FAILED"
       DELIMITED BY SIZE INTO MESG-DESC
       MOVE RET-STATUS TO RP-STATUS
       PERFORM PROCESS-ERROR
       GO TO EXIT-PROGRAM.
```

CDIC MACROS

LIBRARY: SQL
MACRO: CDIC05

```
    IF LOCAL-NULL-FLAG = ZERO
       MOVE ZERO TO TAG-NULL-P1
       MOVE D-P2-P3 TO TAG-P1
    ELSE
       MOVE 1 TO TAG-NULL-P1
       MOVE ZERO TO TAG-P1
       MOVE ZERO TO LOCAL-NULL-FLAG.
```

CDIC MACROS

LIBRARY: SQL
MACRO: CDIC06

```
    IF LOCAL-NULL-FLAG NOT = ZERO
       MOVE ZERO TO TAG-NULL-P1
       MOVE 1 TO TAG-P1
       MOVE ZERO TO LOCAL-NULL-FLAG
       GO TO P2-P3.
```

22.5 CDIMD Retrieve Internal Meta Data

This routine will use a data type name, and access the CDM for its type, size and number of decimal digits.

22.5.1 Inputs

1. Data Type Name

```
01   DATA-TYPE-NAME      PIC X(30).
```

22.5.2 CDM Requirements

1. Data Type - USER_DEF_DATA_TYPE

22.5.3 Internal Requirements

None

22.5.4 Processing

1. Access the CDM entity classes USER_DEF_DATA_TYPE with data type name. Retrieve the type (TYPE_ID), size (MAX_SIZE) and number of decimal digits (NO_OF_DEC) for the specified data type.

2. If the entry is not found, set the error status variable to indicate that an error has occurred.

22.5.5 Outputs

1. Internal Schema Format

```
01  TYPE            PIC X.
01  SIZE            PIC 9(3).
01  ND              PIC 9(2).
```

2. Error Status

```
01  ERROR-STATUS     PIC X(5).
```

22.6 CDMSG Generate Conceptual Schema Search Parameters

This routine will generate the working storage conceptual data definitions required for the runtime update/search values.

22.6.1 Inputs

1. Work File name

```
01  WORK-FILE        PIC X(30).
```

2. Subtransaction Identification

```
01  SUBTRANS-ID          PIC 9(3).
```

3. Conceptual Schema Format

   CS-QUALIFY-LIST
   CS-ACTION-LIST

4. Internal Schema Format

   IS-QUALIFY-LIST
   IS-ACTION-LIST

22.6.2 CDM Requirements

None

22.6.3 Internal Requirements

None

22.6.4 Processing

1. Generate 01 level conceptual schema data definitions for the insert/modify values.

   For each entry in the IS-ACTION-LIST with IS-SUBTRANS-ID equal to the current subtransaction (SUBTRANS-ID) and IS-MAPPED-TO flag set to "Y", generate the following data definition in the WORK-FILE·

   ```
   01   CS-VAR-nn        picture clause.
   ```

   where nn = IS-INDEX.

   Generate the picture clause using values found in the corresponding CS-ACTION-LIST entry.

2. Generate 01 level conceptual schema data definitions for the search parameters.

   For each entry in the IS-QUALIFY-LIST with ISQ-SUBTRANS-IDL equal to the current subtransaction (SUBTRANS-ID) and ISQ-TYPE = 2, generate the following data definition in the WORK-FILE:

   ```
   01   CSQ-VAR-nn        picture clause
   ```

   where nn = ISQ-INDEX.

   Generate the picture clause using values found in the corresponding CSQ-QUALIFY-LIST entry.

3. Generate the start of the linkage section for the request
   processor. Generate the following statements in the CWORK-FILE:

```
LINKAGE-SECTION.
01   MESSAGE-BODY-IN.
   03   CASE-NO      PIC XXX.
   03   SUBID        PIC 999.
```

4. Generate the 03 level data definitions for search parameters contained in the message.

   For each entry in the IS-QUALIFY-LIST with ISQ-SUBTRANS-IDL equal to the current subtransaction (SUBTRANS-ID) and ISQ-TYPE = 2, generate the following data definition in the WORK-FILE:

   03 MSG-VAR-nn picture clause

   where nn = ISQ-INDEX.

   Generate the picture clause using values found in the corresponding CSQ-QUALIFY-LIST entry.

5. Generate the 03 level data definitions for the insert/modify values contained in the message.

   For each entry in the IS-ACTION-LIST with IS-SUBTRANS-ID equal to the current subtransaction (SUBTRANS-ID) and IS-MAPPED-TO flag set to "Y", generate the following data definition in the WORK-FILE:

   03 MSGI-VAR-nn picture clause

   where nn = IS-INDEX.

   Generate the picture clause using values found in the corresponding CS-ACTION-LIST entry.

22.6.5 Outputs

Code will be generated into the working storage and linkage section of the request processor in the following format:

```
01   CS-VAR-nn     PIC    type(size)[V9(nd)].

01   CSQ-VAR-nn    PIC    type(size)[V9(nd)].

LINKAGE SECTION.
          01   MESSAGE-BODY-IN.
               03   CASE-NO  PIC XXX.
               03   SUBID    PIC 999.
               03   MSG-VAR-nn   PIC  type(size)[V9(nd)].
               03   MSGI-VAR-nn  PIC  type(size)[V9(nd)].
```

22.7 CDPIC Generate COBOL Picture Clause

This routine will generate a picture clause data definition for a COBOL identifier according to its type, size and number of decimal digits.

22.7.1 Inputs

1. Format

```
01  TYPE                PIC X.
01  SIZE                PIC 999.
01  ND                  PIC 99.
```

22.7.2 CDM Requirements

None

22.7.3 Internal Requirements

None

22.7.4 Processing

Determine the type for the identifier and generate the appropriate picture clause using the size and number of decimal digits for the identifier. The valid types for an identifier are:

- C - Alphanumeric identifier
- N - Numeric identifier
- P - Computational numeric identifier
- S - Signed numeric identifier
- I - Signed computational numeric identifier
- F - Computational-2 identifier

22.7.5 Output

The generated picture clause will have the following format:

type(size)[V9(nd)][COMP-3].

22.8 CDPRM Generate Internal Schema Search Parameters

This routine will generate the working storage internal data definitions required for the runtime update/search values.

22.8.1 Inputs

1. Work File Name

```
01  WORK-FILE           PIC X(30).
```

2. Subtransaction Identification

```
01  SUBTRANS-ID         PIC 9(3).
```

3. Internal Schema Format

IS-QUALIFY-LIST

4. Complex Mapping Algorithm Table

CMA-TABLE

22.8.2 CDM Requirements

None

22.8.3 Internal Requirements

None

22.8.4 Processing

For each entry in the IS-QUALIFY-LIST with ISQ-SUBTRANS-IDL equal to the current subtransaction (SUBTRANS-ID) and ISQ-TYPE = 2, generate the following data definition in the WORK-FILE.

1. If both ISQ-RTIDL and ISQ-DFIDL do not equal spaces, generate:

```
01  ISQL-VAR-nn          picture clause
```

where nn = ISQ-INDEX.

Call "CDPIC" with ISQ-TYPEL, ISQ-SIZEL and ISQ-NDL to generate a picture clause.

2. If both ISQ-RTIDL and ISQ-DFIDL equal spaces, generate:

```
01  ISQL-VAR-nn          PIC X(30).
```

where nn = ISQ-INDEX.

3. If ISQ-RTIDL does not equal space and ISQ-DFIDL equal space, generate a data definition for each data field parameter of the complex mapping algorithm for this ISQ entry. Determine the entry in the CMA-TABLE containing the module information ISQ-ALG-PTR-L(ISQ-INDEX). For each data field parameter (CMA-DFID NOT EQUAL SPACE) GENERATE:

```
01    ISQL-mod-name-mm-nn     picture clause.
```

where

mod-name = the name of the module for the algorithm (CMA-MOD-ID)
mm = the module instance (CMA-MOD-INST)
nn = the parameter number for the current parameter being generated (CMA-PARM-NO)

Call "CDPIC" with CMA-DF-TYPE, CMA-DF-SIZE and CMA-DF-NO to generate a picture clause.

22.8.5 Outputs

Code will be generatd into the working storage section of the request processor in one of the following formats:

```
01   ISQL-VAR-nn              PIC type(size)[V9(nd)].

01   ISQL-mod-name-mm-nn      PIC type(size)[V9(nd)].
```

22.9 CDQDF Generate Internal Schema Retrieval Qualification Variables

This routine will generate the working storage internal data definitions for the data fields that will be used for retrieval/update qualifications. It will also generate null flag variables for each internal data definition generated.

22.9.1 Inputs

1. Work File Name

```
01  WORK-FILE           PIC X(30).
```

2. Subtransaction Identification

```
01  SUBTRANS-ID         PIC 9(3).
```

3. Internal Schema Format

IS-QUALIFY-LIST

4. Complex Mapping Algorithm Table

CMA-TABLE

22.9.2 CDM Requirements

None

22.9.3 Internal Requirements

None

22.9.4 Processing

Generate the 01 level data definition for the qualification variables.

For each half of an entry in the IS-QUALIFY-LIST with ISQ-TYPE = 3 and ISQ-SUBTRANS-IDL or ISQ-SUBTRANS-IDR equal to the current subtransaction, generate the following data definition in the WORK-FILE.

1. If the entry is from the left half of the IS-QUALIFY-LIST and both ISQ-RTIDL and ISQ-DFIDL do not equal spaces, generate:

```
01  ISQL-VAR-nn          picture clause
01  ISQL-NULL-nn         PIC 9.
```

If the entry is from the right half of the IS-QUALIFY-LIST and both ISQ-RTIDR and ISQ-DFIDR do not equal spaces, generate:

```
01  ISQR-VAR-nn          picture clause
01  ISQR-NULL-nn         PIC 9.
```

where nn = ISQ-INDEX.

Call "CDPIC" with ISQ-TYPE(L/R), ISQ-SIZE(L/R) and ISQ-ND(L/R) to generate a picture clause.

2. If the entry is from the left half of the IS-QUALIFY-LIST and ISQ-RTIDL and both ISQ-DFIDL equal spaces, generate:

```
01  ISQR-VAR-nn            PIC X(30).
```

where nn = ISQ-INDEX.

If the entry is from the right half of the IS-QUALIFY-LIST and both ISQ-RTIDR and ISQ-DFIDR equal spaces, generate:

```
01  ISQR-VAR-nn            PIC X(30).
```

where nn = ISQ-INDEX.

3. If the entry is from the left half of the IS-QUALIFY-LIST and ISQ-RTIDL does not equal space and ISQ-DFIDL equal space, generate a data definition for each data field parameter of the complex mapping algorithm (if they have not already been generated). Determine the entry in the CMA-TABLE containing the module information ISQ-ALG-PTR-L(ISQ-INDEX). For each data field parameter (CMA-DFID not equal space) generate:

```
01   ISQL-mod-name-mm-nn  picture clause
```

where

```
mod-name = the name of the module for the algorithm
           (CMA-MOD-ID)
mm       = the module instance (CMA-MOD-INST)
nn       = the parameter number for the current
           parameter being generated (CMA-PARM-NO)
```

Call "CDPIC" with CMA-DF-TYPE, CMA-DF-SIZE and CMA-DF-NO to generate a picture clause.

If the entry is from the right half of the IS-QUALIFY-LIST and ISQ-RTIDR does not equal space and ISQ-DFIDR equal space, generate a data definition for each data field parameter of the complex mapping algorithm (if they have not already been generated).

Determine the entry in the CMA-TABLE containing the module information ISQ-ALG-PTR-R(ISQ-INDEX). For each data field parameter (CMA-DFID not equal space) generate:

```
01   ISQR-mod-name-mm-nn     picture clause
```

where

mod-name = the name of the module for the algorithm (CMA-MOD-ID)

22.9.5 Output

Code will be generated into the working storage section of the request processor in one of the following formats:

```
01  ISQL-VAR-nn             PIC type(size)[V9(nd)].
01  ISQR-VAR-nn             PIC type(size)[V9(nd)].
01  ISQL-mod-name-mm-nn     PIC type(size)[V9(nd)].
01  ISQR-mod-name-mm-nn     PIC type(size)[V9(nd)].
01  ISQL-NULL-nn            PIC 9.
01  ISQR-NULL-nn            PIC 9.
```

22.10 CDRDF Generate Internal Schema Retrieval Data Fields

This routine will generate the working storage internal data definitions for the retrieved data fields that will be used in converting internal format to conceptual format.

22.10.1 Inputs

1. Work File Name

```
01  WORK-FILE           PIC X(30).
```

2. Subtransaction Identification

```
01  SUBTRANS-ID         PIC 9(3).
```

3. Internal Schema Format

IS-ACTION-LIST

4. Complex Mapping Algorithm Table

CMA-TABLE

22.10.2 CDM Requirements

None

22.10.3 Internal Requirements

None

22.10.4 Processing

For each entry in the IS-ACTION-LIST with IS-SUBTRANS-ID equal to the current subtransaction (SUBTRANS-ID), generate the following data definition in the WORK-FILE.

1. If both IS-RTID and IS-DFID do not equal spaces, generate:

```
01  IS-VAR-nn              picture clause.
01  INDP-nn                PIC S9(4) COMP.
```

where nn = IS-INDEX

Call "CDPIC" with IS-DATA-TYPE, IS-SIZE and IS-ND to generate the picture clause.

2. If both IS-RTID and IS-DFID equal spaces, generate:

```
01  IS-VAR-nn              PIC X(30).
01  INDP-nn                PIC S9(4) COMP.
```

where nn = IS-INDEX

3. If IS-RTID does not equal space and IS-DFID equal space, generate a data definition for each data field parameter of the complex mapping algorithm for this IS entry. Determine the entry in the CMA-TABLE containing the module information IS-ALG-PTR (IS-INDEX). For each data field parameter (CMA-DFID not equal space) generate:

```
01  ISQ-mod-name-mm-nn        picture clause.
01  INDP-mod-name-mm-nn       PIC S9(4) COMP.
```

where

mod-name = the name of the module for the algorithm (CMA-MOD-ID)
mm = the module instance (CMA-MOD-INST)
nn = the parameter number for the current parameter being generated (CMA-PARM-NO)

Call "CDPIC" with CMA-DF-TYPE, CMA-DF-SIZE and CMA-DF-NO to generate a picture clause.

22.10.5 Outputs

Code will be generated into the working storage section of the request processor in the following format:

```
01  IS-VAR-nn             PIC type(size)[V9(nd)].
01  INDP-nn               PIC S9(4) COMP.
01  IS-mod-name-mm-nn     PIC type(size)[V9(nd)].
01  INDP-mod-name-nn-nn   PIC S9(4) COMP.
```

22.11 CDRFT Generate Conceptual Schema Retrieval Data Fields

This routine will generate the FD data definitions for the results file of retrieved data and working storage data definitions for the retrieved data. This must be the last called routine to generate File Section and the beginning of working storage.

22.11.1 Inputs

1. Work File Name

   01 WORK-FILE PIC X(30).

2. Subtransaction Identifcation

   01 SUBTRANS-ID PIC 9(3).

3. Result Field Table

   RFT

22.11.2 CDM Requirements

None

22.11.3 Internal Requirements

None

22.11.4 Processing

1. Generate the 01 level for the record description. Generate the following statement in the WORK-FILE:

   01 RESULTS-REC.

1a Generate the 03 level data definitions for the null flag indicators of the results.

For each entry in the RFT with RFT-SUBTRANS equal to the current subtransaction (SUBTRANS-ID), generate the following data definitions in the WORK-FILE:

03 RES-NULL-nn PIC 9.

where nn = RFT-IS-PTR

2. Generate the 03 level data definitions for the record description.

For each entry in the RFT with RFT-SUBTRANS equal to the current subtransaction (SUBTRANS-ID), generate the following data definitions in the WORK-FILE:

03 RES-nn picture clause

where nn = RFT-IS-PTR

Call "CDPIC" with RFT-SIZE, RFT-TYPE and RFT-ND to generate the picture clause.

3. Generate the start of working storage. Generate the following statement in the WORK-FILE:

   WORKING-STORAGE SECTION.

4. Generate 01 level data definitions for the retrieved data fields.

   For each entry in the RFT with RFT-SUBTRANS equal to the current subtransaction (SUBTRANS-ID), generate the following data definitions in the WORK-FILE:

   01 CS-VAR-nn picture clause

   where nn = RFT-IS-PTR

   Call "CDPIC" with RFT-SIZE, RFT-TYPE and RFT-ND to generate the picture clause.

22.11.5 Outputs

Code will be generated into the File Section and the Working Storage Section of the request processor in the following format:

```
01  RESULTS-REC.
03  RES-NULL-nn        PIC 9.
03  RES-nn             PIC type(size)[V9(nd)].

  WORKING-STORAGE SECTION.
01  CS-VAR-nn          PIC type(size)[V9(nd)].
```

22.12 Macro Expander

This subprogram will generate code as does a macro expander into the named file. The actual macro definitions are found in the CDM database.

22.12.1 Inputs

1. Output File Name

   01 FILE-NAME PIC X(30).

   File to which the macro code will be expanded.

2. MACRO-NAME

   01 MACRO-NAME PIC X(8).

   The name of the macro to be expanded.

3. Substitution List

A list containing the substitution parameters and values to be used in the macro expansion.

4. RET-STATUS

```
01   RET-STATUS      PIC X(5).
```

22.12.2 CDM Requirements

None

22.12.3 Internal Requirements

None

22.12.4 Processing

1. Open the CDM and retrieve the macro lines from the CDM database.

2. Perform substitution of parameters with the values specified in the call.

   2.1 Move the substitution parameters to the compare area.

   2.2 If the compare string is not empty, then start to do the comparison and replacement until the WORK-LINE is exhausted or the new line is full.

   2.3 If the window position has not reached the end of the WORK-LINE and there is still room in the new line, then close the window by moving the rest of the WORK-LINE to the new line.

   2.4 Move the new line to the WORK-LINE.

3. Compare each column in WORK-LINE with each column in the substitution parameter. If not equal, move the column of WORK-LINE to MACRO-DATA and reset the match count to zero, increment the WORK-LINE and MACRO-DATA pointers by 1. If equal, then increment the match count and check if the compare string needs to wrapped around to compare with the WORK-LINE. If the match count reaches the length of the compare string, which means there is an identical substring in the WORK-LINE, then move the substitution new value to the MACRO-DATA and update the pointers in both WORK-LINE and MACRO-DATA.

22.12.5 Outputs

The macro will be expanded into the output file a line at a time from WORK-LINE.

```
01   WORK-LINE       PIC X(72).
```

22.13 Function CDCMPRM Generate Complex Mapping Algorithm Parameters

This routine will generate the working storage data definitions for the Complex Mapping Algorithm Parameters.

22.13.1 Inputs

1. Work File Name

   01 WORK-FILE PIC X(30)

2. Complex Mapping Algorithm Table

   CMA-TABLE

3. Subroutine Identification

   01 SUBTRANS-ID PIC 9(3)

22.13.2 CDM Requirements

None

22.13.3 Internal Requirements

None

22.13.4 Processing

1. Generate the data definitions for the Complex Mapping Algorithm Parameters.

   For each entry in the CMA-TABLE with CMA-SUBTRANSACTION equal to the current subtransaction (SUBTRANS-ID), generate the following data definition in the WORK-FILE for each parameter of the algorithm.

   01 PARM-mod-name-mm-nn picture clause

   where

   mod-name = the name of the module for the Algorithm (CMA-MOD-ID)
   mm = the module instance (CMA-MOD-INST)
   nn = the parameter number for the current parameter being generated (CMA-PARM-NO)

   Call routine "CDPIC" with the values in CMA-TABLE for type, size, and number of decimals of the parameter (CMA-PARM-TYPE, CMA-PARM-SIZE, CMA-PARM-ND) to generate the picture clause.

22.13.5 Outputs

1. Code will be generated into the working storage section of the request processor in the following format:

   01 PARM-mod-name-mm-nn PIC type(size)[v9(nd)].

2. Return Status

```
01    RETURN-STATUS        PIC X(5).
```

22.14 Function CDGENRT Generate A Record Structure

This function will generate a COBOL working storage record layout for a specified record type.

22.14.1 Inputs

1. Internal Schema Definition of record.

   DB-ID
   RT-ID

2. Work File Name

   WORK-FILE-NAME

22.14.2 CDM Requirements

None

22.14.3 Internal Requirements

1. Table to hold the entire record structure for a specified record.

   TDFT-TABLE include file in IISSCLIB

22.14.4 Processing

1. Initialize all rows in the data field table (TDFT-TABLE).

2. Populate the data field table with the data base and record type identification information for the record structure to be generated.

   TDFT-DBID = DB-ID
   TDFT-RTID = RT-ID

3. Call routine "RETRFLD" to populate the data field table with the entire record structure with the following parameters:

   TDFT-TABLE
   MODULE-STATUS

4. Initialize all local variables used to build the various clauses of the data field definition.

5. Process the data field table for the record. For each used TDFT-DFID, perform steps 5.1 - 5.10.

5.1 Bypass an index field that was generated by the system.

```
If TDFT-INDEX-IND = "G"
continue processing at step 5.
```

5.2 Increment the data field level number to obtain the COBOL data definition level.

```
COBOL-LEVEL = TDFT-LEVEL (TDFT-INDEX) + 2
```

5.3 Construct the picture clause for an elementary data field that is not a filler.

5.3.1 If TDFT-DATA-TYPE-NAME does not equal zeroes and TDFT-FILLER-SIZE equal zero, call routine "CDIMD" with the following parameters to obtain the type, size and number of decimals for this data field.

```
TDFT-DATA-TYPE-NAME
TYPE
SIZE
NO-DECIMALS
MODULE-STATUS
```

Call routine "CDPIC" with the following parameters retrieved from routine to construct a COBOL picture clause.

```
TYPE
SIZE
NO-DECIMALS
PICTURE-CLAUSE
```

Continue processing at step 5.6.

5.4 Construct the picture clause for a filler data field.

5.4.1 If TDFT-DATA-TYPE equal zeroes and TDFT-FILLER-SIZE greater than zero, set

```
TYPE = "C"
SIZE = TDFT-FILLER-SIZE
NO-DECIMALS = 0
```

Call routine "CDPIC" with the following parameters established above, to construct a COBOL picture clause.

```
TYPE
SIZE
NO-DECIMALS
PICTURE-CLAUSE
```

Continue processing at step 5.6.

5.5 Construct the picture clause for a group data field.

5.5.1 If TDFT-DATA-TYPE equal zeroes and TDFT-FILLER-SIZE equal zero, set PICTURE-CLAUSE = spaces

5.6 Construct the REDEFINES clause.

5.6.1 Initialize the REDEFINES clause. Set REDEFINES-CLAUSE = spaces.

5.6.2 If TDFT-REDEF-DF-NO does not equal zero, construct the redefines clause. Set REDEFINES-CLAUSE = 'REDEFINES D-dfno'

where

dfno = TDFT-REDEF-DF-NO

5.7 Construct the OCCURS clause.

5.7.1 Initialize the OCCURS clause. Set

OCCURS-CLAUSE = spaces.

5.7.2 If TDFT-OCCURS value is greater than zero, construct the OCCURS clause. Set

OCCURS-CLAUSE = 'OCCURS nn TIMES'

where

nn = TDFT-OCCURS

5.8 Construct the DEPENDING clause.

5.8.1 Initialize the DEPENDING clause.

DEPENDING-CLAUSE = spaces

5.8.2 If TDFT-OCC-DEP-DF does not equal zero, construct the DEPENDING clause. Set

DEPENDING-CLAUSE = 'DEPENDING ON D-dfno'

where

dfno = TDFT-OCC-DEP-DF

5.9 Construct the INDEX clause.

5.9.1 Initialize the INDEX clause.

INDEX-CLAUSE = spaces

5.9.2 If TDFT-INDEXED-DF does not equal zero, construct the INDEX clause. Set

INDEX-CLAUSE = 'INDEXED BY INDEX-dfno'

where

dfno = TDFT-DFNO

5.10 Generate the definition for the data field. Generate the following:

```
COBOL-LEVEL   D-dfno
              REDEFINES-CLAUSE
              OCCURS-CLAUSE
              DEPENDING-CLAUSE
              INDEX CLAUSE
              PICTURE-CLAUSE
                .
```

where

dfno = TDFT-DFNO

REDEFINES-CLAUSE, OCCURS-CLAUSE, PICTURE-CLAUSE, DEPENDING-CLAUSE, INDEX-CLAUSE are constructed in steps 5.3 - 5.9 and are not to be used if the local variable contains a value of spaces.

COBOL-LEVEL is constructed in step 5.2.

5.11 Generate the data definition for the variable to hold the maximum value of a field that is an index for a repeating data field.

Search each used entry in the data field table (TDFT-TABLE) for entries that are an index for a repeating data field. If TDFT-INDEXED-DF does not equal zero, generate the following:

```
01    INDEX-dfno-MAX          PIC S9(5) COMP.
```

where

dfno = TDFT-DFNO

22.14.5 Outputs

1. Code will be generated into the working storage section of the request processor in the following format:

```
01    T-rtno.
      03    D-dfno          picture-clause
      03    D-dfno          picture-clause
       .
       .
       .

01    INDEX-dfno-MAX        PIC S9(5) COMP.
```

2. Module status of the function, indicating success or errors.

22.15 Function CDGENIF

This function will generate a COBOL IF Statement that will evaluate user qualifications contained in an NDML WHERE clause at the conceptual level in the CS/ES transfer program.

22.15.1 Inputs

1. Boolean Operators, conditions and parenthesis from the NDML WHERE clause.

   BOOLEAN-LIST

2. Conceptual Schema Representation of the NDML WHERE clause.

   CS-QUALIFY-LIST
   CS-ACTION-LIST

3. Internal Schema Representation of the NDML WHERE clause.

   IS-QUALIFY-LIST

4. Name of the file where the IF Statement is to be generated.

   01 FILE-NAME PIC X(30)

22.15.2 CDM Requirements

None

22.15.3 Internal Requirements

None

22.15.4 Processing

1. Generate the start of an IF Statement in the specified file. Generate:

   IF

2. Generate a Conceptual Schema IF Statement. Process the entire BOOLEAN-LIST. Perform steps 2.1 - 2.3 until BL-INDEX > BL-USED.

   2.1 Process an operator or parenthesis from the BOOLEAN-LIST. Generate into specified file, if BL-OP (BL-INDEX) not equal space:

   oper/paren

   where

   oper/paren = BL-OP (BL-INDEX)

Continue processing at step 2.1

2.2 Process a condition in the BOOLEAN-LIST. Generate into the specified file, if BL-CSQ-PTR not equal zero and CSQ-OP(BL-CSQ-PTR(BL-INDEX)) not = "NN" or "NL":

```
(CS-VARcc op CSQ-VAR-qq AND
 CS-NULL-FLAG-cc = 0)
```

where

```
cc = BL-CS-PTR(BL-INDEX)
op = CSQ-OP(BL-CSQ-PTR(BL-INDEX))
qq = BL-CSQ-PTR(BL-INDEX)
```

Continue processing at step 2.1

2.3 Process an IS NULL or IS NOT NULL condition in the BOOLEAN-LIST. Generate into the specified file, if BL-CSQ-PTR not equal zero and CSQ-OP(BL-CSQ-PTR(BL-INDEX)) = "NN" or "NL":

```
CS-NULL-FLAG-cc = value
```

where

```
cc    = BL-CS-PTR(BL-INDEX)
value = 1 if CSQ-OP(BL-CSQ-PTR(BL-INDEX)) = NL
        0 if CSQ-OP(BL-CSQ-PTR(BL-INDEX)) = NN
```

Continue processing at step 2.1

2.4 Generate the remainder of the conceptual schema IF statement for type 3 qualifications (column op column)

2.4.1 Generate the start of the additional qualification. Generate the following in the specified file:

```
"AND ( "
```

2.4.2 Generate the following in the specified file for each entry in the IS-QUALIFY-LIST where:

```
ISQ-EVAL-FLAG (ISQ-INDEX) = 0
ISQ-TYPE (ISQ-INDEX) = 3

(CS-VARc1 op CS-VARcr AND
 CS-NULL-FLAG-c1 = 0 AND CS-NULL-FLAG-cr = 0)
     operator
```

where:

c1 = entry in the CS-ACTION-LIST with matching TAG number from the CSQ-AUCL (ISQ-CSQ-PTR (ISQ-INDEX)) entry.
op = CSQ-OP (ISQ-CSQ-PTR (ISQ-INDEX))

cr = entry in the CS-ACTION-LIST with matching TAG number from the CSQ-AUCR (ISQ-CSQ-PTR (ISQ-INDEX)) entry.
operator = AND if not the last ISQ entry blank if the last ISQ entry.

2.4.3 Generate the close of the additional qualification. Generate the following in the specified file:

")"

3. Exit processing.

22.15.5 Outputs

1. Status of function processing.

RET-STATUS PIC X(5).

2. Generated IF Statement on the specified file.

22.16 CDGTV - Generate Tag Variable Definitions

Function CDGTV generates working storage tag variable and indicator definitions in support of conceptual evaluation in the presence of complex internal to conceptual mapping for deletes and modifies.

22.16.1 Inputs

1. CMA-TABLE
2. IS-QUALIFY-LIST
3. CS-QUALIFY-LIST
4. WORK-FILE
5. SUBTRANS-ID

22.16.2 CDM Requirements

None

22.16.3 Internal Requirements

None

22.16.4 Processing

1. Open WORK-FILE for EXTEND.
2. Scan the IS-QUALIFY-LIST. If all used ISQ entries have ISQ-EVAL-FLAG not equal zero for all entries with either ISQ-SUBTRANS-IDL or ISQ-SUBTRANS-IDR equal SUBTRANS-ID, go to step 7.

3. Scan the IS-QUALIFY-LIST. For each used with ISQ-SUBTRANS-IDL equal SUBTRANS-ID and ISQ-ALG-PTR-L equal zero, generate the following two lines if not previously generated.

```
01 TAG-tno        pic clause.
01 TAG-NULL-tno   PIC 9.
```

where tno is the CSQ-AUCL value pointed to by the current ISQ-CSQ-PTR and pic clause is the value generated by calling CDPIC with the CSQ-L-TYPE, CSQ-L-SIZE and CSQ-L-ND pointed to by the current ISQ-CSQ-PTR.

4. Scan the IS-QUALIFY-LIST. For each used with ISQ-SUBTRANS-IDR equal SUBTRANS-ID and ISQ-DFNOR not equal zero and ISQ-ALG-PTR-R equal zero, generate the following two lines, if not previously generated.

```
01 TAG-tno        pic clause.
01 TAG-NULL-tno   PIC 9.
```

where tno is the CSQ-AUCR value pointed to by the current ISQ-CSQ-PTR and pic clause is the value generated by calling CDPIC with the CSQ-R-TYPE, CSQ-R-SIZE and CSQ-R-ND pointed to by the current ISQ-CSQ-PTR.

5. Scan the IS-QUALIFY-LIST. For each used ISQ left entry with ISQ-SUBTRANS-IDL equal SUBTRANS-ID and ISQ-ALG-PTR-L not equal zero, process the CMA-ALG-ENTRY pointed to by ISQ-ALG-PTR-L as follows.

   For each CMA-PARM-ENTRY with a non-zero CMA-TAG-NO, which was not previously generated, generate

```
01 TAG-tno        pic clause.
01 TAG-NULL-tno   PIC 9.
```

   where tno is the CMA-TAG-NO value and pic clause is generated by calling CDPIC with the CSQ-L-TYPE, CSQ-L-SIZE and CSQ-L-ND where the CMA-TAG-NO matches the first CSQ-AUCL or CSQ-AUCR.

6. Scan the IS-QUALIFY-LIST. For each used ISQ right entry with a non-zero ISQ-RTNOR and ISQ-SUBTRANS-IDR equal SUBTRANS-ID and ISQ-ALG-PTR-R not equal zero, process the CMA-ALG-ENTRY pointed to by ISQ-ALG-PTR-R as follows.

   For each CMA-PARM-ENTRY with a non-zero CMA-TAG-NO, which was not previously generated, generate

```
01 TAG-tno        pic clause.
01 TAG-NULL-tno   PIC 9.
```

   where tno is the CMA-TAG-NO value and pic clause is generated by calling CDPIC with the CSQ-R-TYPE, CSQ-R-SIZE and CSQ-R-ND where the CMA-TAG-NO matches the first CSQ-AUCL or CSQ-AUCR.

7. Close WORK-FILE and exit.

22.16.5 Outputs

Code will be generated in the specified file in the following format:

```
01 TAG-tno        pic clause.
01 TAG-NULL-tno   PIC 9.
```

22.17 Function CDGDF - Generate Datafield and Indicator Variables

Function CDGDF generates working storage variable definitions and indicators for receiving fields into a SQL request subroutine.

22.17.1 Inputs

1. IS-ACTION-LIST
2. IS-QUALIFY-LIST
3. CMA-TABLE
4. SUBTRANS-ID
5. WORK-FILE

22.17.2 CDM Requirements

None

22.17.3 Internal Requirements

None

22.17.4 Processing

1. Open WORK-FILE for EXTEND.
2. For each unique used IS-ACTION entry with both IS-DFNO and IS-RTNO not equal zero and IS-SUBTRANS-ID equal SUBTRANS-ID, generate the following two lines.

   ```
   01 D-dfno-rtno    pic clause.
   01 INDP-dfno-rtno PIC S9(4) COMP.
   ```

   where dfno is the IS-DFNO value, rtno is the IS-RTNO value and pic clause is generated by calling CDPIC with IS-TYPE, IS-SIZE and IS-ND.
3. For each unique non-zero ISQ-DFNOL/ISQ-RTNOL or ISQ-DFNOR/ISQ-RTNOR combination with the corresponding ISQ-SUBTRANS-IDL or ISQ-SUBTRANS-IDR equal SUBTRANS-ID which has not previously been generated, generate

```
01 D-dfno-rtno     pic clause.
01 INDP-dfno-rtno PIC S9(4) COMP.
```

where dfno is either the ISQ-DFNOL or ISQ-DFNOR value, rtno is either the ISQ-RTNOL or ISQ-RTNOR value and pic clause is generated by calling CDPIC with either ISQ-TYPEL, ISQ-SIZEL and ISQ-NDL or ISQ-TYPER, ISQ-SIZER and ISQ-NDR.

4. For each CMA-ALG-ENTRY pointed to by either a used ISQ-ALG-PTR-L or ISQ-ALG-PTR-R with its corresponding ISQ-SUBTRANS-IDL or ISQ-SUBTRANS-IDR equal SUBTRANS-ID, perform the following steps.

   Search the CMA-PARM-ENTRYs looking for all non-zero CMA-DFNO/CMA-RT-NO combinations. For each such entry, generate the following two lines, if an identical entry has not been previously generated.

```
01 D-dfno-rtno     pic clause.
01 INDP-dfno-rtno PIC S9(4) COMP.
```

   where dfno is the CMA-DF-NO value, rtno is the CMA-RT-NO value and pic clause is generated by CDPIC using CMA-DF-TYPE, CMA-DF-SIZE and CMA-DF-ND.

   Search the CMA-PARM-ENTRYs looking for a non-zero CMA-RTNO and a CMA-DF-NO equal zero. If found, generate the following two lines for each unique CMA-RT-NO/used DF-DFNO combination not previously generated.

```
01 D-dfno-rtno     pic clause.
01 INDP-dfno-rtno PIC S9(4) COMP.
```

   where dfno is the DF-DFNO value, rtno is the CMA-RT-NO value and pic clause is generated by calling CDPIC with DF-TYPE, DF-SIZE and DF-ND.

5. Close WORK-FILE and exit.

22.17.5 Outputs

Code will be generated in the specified file in the following format:

```
01 D-dfno-rtno          pic clause.

01 INDP-dfno-rtno       PIC S9(4) COMP.
```

22.18 Function CDGNV - Generate User-Defined NULL Variable Names

This function generates user-defined NULL variable names and picture clauses into the SQL request processor working storage section. The variables generated by this program will participate in the SQL WHERE clause.

22.18.1 Inputs

1. IS-QUALIFY-LIST
2. SUBTRANS-ID
3. NUMERIC-NULL
4. CHAR-NULL
5. FILE-NAME

22.18.2 CDM Requirements

None

22.18.3 Internal Requirements

None

22.18.4 Processing

1. Open WORK-FILE for EXTEND.
2. If NUMERIC-NULL not equal NULL, generate

   01 DB-NUM-NULL PIC S9(9)V9(9) COMP-3.

3. If CHAR-NULL not equal NULL, generate a variable definition as follows for each uniquely sized non-complex entry with either ISQ-TYPEL or ISQ-TYPER equal C and whose respective ISQ-SUBTRANS-IDL or ISQ-SUBTRANS-IDR equal SUBTRANS-ID.

   01 DB-CHAR-NULL-nn PIC X(nn).

   where nn is either the ISQ-SIZEL or ISQ-SIZER value.

   Generate no duplicates.

4. Close WORK-FILE and exit.

22.18.5 Outputs

Code will be generated in the specified file in the following format:

```
01 DB-NUM-NULL              PIC S9(9)V9(9).

01 DB-CHAR-NULL-nn          PIC X(nn).
```

22.19 Function CDRPCIF - Generate COBOL IF Statement for Conceptual Schema

This function will generate a COBOL IF statement that will evaluate user qualifications contained in an NDML WHERE clause. The IF statement will be generated into a request processer sub-program to perform the evaluation of the WHERE

clause at the conceptual schema level. This IF statement will be necessary for any update transactions that contained complex mapping algorithms in the WHERE clause.

22.19.1 Inputs

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

   BOOLEAN-LIST

2. Conceptual Schema Representation of the NDML WHERE clause.

   CS-QUALIFY-LIST
   CS-ACTION-LIST

3. Internal Schema Representation of the NDML WHERE clause.

   IS-QUALIFY-LIST

4. Name of the file where the IF statement is to be generated.

   01 FILE-NAME PIC X(30).

22.19.2 CDM Requirements

None

22.19.3 Internal Requirements

None

22.19.4 Processing

1. Generate the start of an IF statement in the specified file. Generate:

   IF

2. Generate a Conceptual Schema IF Statement. Process the entire BOOLEAN-LIST. Perform steps 2.1 - 2.3 until BL-INDEX > BL-USED.

   2.1 Process an operator or parenthesis from the BOOLEAN-LIST. Generate into specified file, if BL-OP (BL-INDEX) not equal space:

      oper/paren

   where

      oper/paren = BL-OP (BL-INDEX)

   Continue processing at step 2.1

2.2 Process a condition in the BOOLEAN-LIST. Generate into the specified file, if BL-CSQ-PTR not equal zero and CSQ-OP(BL-CSQ-PTR(BL-INDEX)) not = "NN" or "NL":

(TAG-tagno op MSG-VAR-nn AND tag-null-tagno = 0)

where

tagno = CS-AUC (BL-CS-PTR (BL-INDEX))
op = CSQ-OP (BL-CSQ-PTR (BL-INDEX))
nn = ISQ-INDEX with ISQ-BL-PTR equal to the current BL-INDEX

Continue processing at step 2.1.

2.3 Process an IS NULL or IS NOT NULL condition in the BOOLEAN-LIST. Generate into the specified file, if BL-CSQ-PTR not equal zero and CSQ-OP(BL-CSQ-PTR(BL-INDEX)) = "NN" or "NL".

TAG-NULL-FLAG-cc = value

where

cc = BL-CS-PTR(BL-INDEX)
value = 1 if CSQ-OP(BL-CSQ-PTR(BL-INDEX)) = NL
0 if CSQ-OP(BL-CSQ-PTR(BL-INDEX)) = NN

Continue processing at step 2.1.

2.4 Generate the remainder of the Conceptual Schema IF statement for Type 3 qualifications (column op column).

2.4.1 Generate the start of the additional qualification. Generate the following in the specified file:

"AND ("

2.4.2 Generate the following in the specified file for each entry in the IS-QUALIFY-LIST where:

ISQ-EVAL-FLAG (ISQ-INDX) = 0
ISQ-TYPE (ISQ-INDEX) = 3
ISQ-SUBTRANS-IDL (ISQ-INDEX) = SUBTRANS-ID

(TAG-tagno1 op TAG-tagno2 AND TAG-NULL-tagno1 = 0 AND TAG-NULL-tagno2 = 0) operator

where

tagno1 = CSQ=-AUCL (ISQ-CSQ-PTR (ISQ-INDEX))
op = CSQ-OP (ISQ-CSQ-PTR (ISQ-INDEX))
tagno2 = CSQ-AUCR (ISQ-CSQ-PTR (ISQ-INDEX))

operator = AND if not the last ISQ entry blank if the last ISQ entry

2.4.3 Generate the close of the additional qualification. Generate the following in the specified file:

")"

22.19.5 Outputs

1. Status of function processing.

RET-STATUS PIC X(5).

2. Generated IF Statement on the specified file.

22.20 Function CDRPIIF - Generate COBOL IF Statement for Internal Schema

This function will generate a COBOL IF Statement that will evaluate user qualifications contained in an NDML WHERE clause. The IF statement will be generated into a request processor sub-program to perform the evaluation of the WHERE clause at the internal schema level. This IF statement will be necessary for any update transactions that contain qualification.

22.20.1 Inputs

1. Boolean Operators, conditions and parenthesis from the NDML WHERE clause.

SUBTRANS-BOOLEAN-LIST

2. Internal Schema Representation of the NDML WHERE clause.

IS-QUALIFY-LIST

3. Name of the file where the IF Statement is to be generated.

```
01  FILE-NAME  PIC X(30)
```

4. Subtransaction Identification.

```
01  SUBTRANS-ID  PIC 9(3)
```

5. Data Base user defined null values.

```
01  CHARACTER-NULL  PIC X(30).
01  NUMERIC-NULL    PIC X(30).
```

22.20.2 CDM Requirements

None

22.20.3 Internal Requirements

None

22.20.4 Processing

1. Generate the start of an IF Statement in the specified file. Generate:

   IF

2. Generate an Internal Schema IF Statement. Process the portion of the SUBTRANS-BOOLEAN-LIST for the specified subtransaction. Perform steps 2.1 - 2.3 for each entry in the SUBTRANS-BOOLEAN-LIST where

   SBL-SUBTRANS(SL-INDEX) = SUBTRANS-ID

   2.1 Process an operator or parenthesis from the SUBTRANS-BOOLEAN-LIST. Generate into the specified file, if SBL-OP(SBL-INDEX) not equal space

   oper/paren

   where

   oper/paren = SBL-OP (SBL-INDEX)

   Continue processing at step 2.

   2.2 Process an external qualification condition in SUBTRANS-BOOLEAN-LIST. Generate into the specified file, if SBL-ISQ-PTR not equal zero and ISQ-OP(SBL-ISQ-PTR(SBL-INDEX)) not = "NN" or "NL" and ISQ-TYPE(SBL-ISQ-PTR(SBL-INDEX)) = "2" and SOURCE-IS-EXTERNAL(SBL-ISQ-PTR(SBL-INDEX))

   (D-dfno op ISQ-VAR-qq AND
   D-dfno NOT = null-value)

   where

   dfno = ISQ-DFNOL(SBL-ISQ-PTR(SBL-INDEX))
   op = ISQ-OP(SBL-ISQ-PTR(SBL-INDEX))
   qq = SBL-ISQ-PTR(SBL-INDEX)

   IF ISQ-TYPE(SBL-ISQ-PTR(SBL-INDEX)) = "C"

   null-value = CHARACTER-NULL

   ELSE

   null-value = NUMERIC-NULL

   Continue processing at step 2.1.

   2.3 Process an IS NULL or IS NOT NULL external qualification condition in the SUBTRANS-BOOLEAN-LIST. Generate into the specified file, if SBL-ISQ-PTR not equal zero and ISQ-OPM(SBL-ISQ-PTR(SBL-INDEX)) = "NN" or "NL", and ISQ-TYPE(SBL-ISQ-PTR(SBL-INDEX)) = "2" and SOURCE-IS-EXTERNAL(SBL-ISQ-PTR(SBL-INDEX))

```
   D-dfno op null-value

where

   dfno = ISQ-DFNOL(SBL-ISQ-PTR(SBL-INDEX))

IF ISQ-TYPEL(SBL-ISQ-PTR(SBL-INDEX)) = "C"

   null-value = CHARACTER-NULL

ELSE

   null-value = NUMERIC-NULL

IF ISQ-OP(SBL-ISQ-PTR(SBL-INDEX)) = "NN"

   op = "="

ELSE

   op = "NOT="

Continue processing at step 2.
```

2.4 Process a union qualification in SUBTRANS-BOOLEAN-LIST. Generate into the specified file if:

```
   SBL-ISQ-PTR(SBL-INDEX) NOT = ZERO AND
   ISQ-TYPE(SBL-ISQ-PTR(SBL-INDEX)) = "2" AND
   SOURCE-IS-UNION(SBL-ISQ-PTR(SBL-INDEX))

   (D-dfno op value AND
    D-dfno NOT = null-value)

where

   dfno = ISQ-DFNOL(SBL-ISQ-PTR(SBL-INDEX))
   op   = ISQ-OP(SBL-ISQ-PTR(SBL-INDEX))

If ISQ-TYPEL(SBL-ISQ-PTR(SBL-INDEX)) = "C"

   value = "ISQ-UNION-VALUE(SBL-ISQ-PTR(SBL-INDEX))"
   null-value = CHARACTER-NULL

ELSE

   value = ISQ-UNION-VALUE(SBL-ISQ-PTR(SBL-INDEX))
   null-value = NUMERIC-NULL

Continue processing at step 2.1.
```

2.5 Process an intra-subtransaction qualification in SUBTRANS-BOOLEAN-LIST. Generate into the specified file if:

```
   SBL-ISQ-PTR(SBL-INDEX) NOT = ZERO AND
   ISQ-TYPE(SBL-ISQ-PTR(SBL-INDEX)) = "3"
```

```
(D-dfno1 op D-dfno2 AND
 D-dfno1 NOT = null-value1 AND
 D-dfno2 NOT = null-value2)

where

   dfno1 = ISQ-DFNOL(SBL-ISQ-PTR(SBL-INDEX))
   op    = ISQ-OP(SBL-ISQ-PTR(SBL-INDEX))
   dfno2 = ISQ-DFNOR(SBL-ISQ-PTR(SBL-INDEX))

IF ISQ-TYPEL(SBL-ISQ-PTR(SBL-INDEX)) = "C"

   null-value1 = CHARACTER-NULL

ELSE

   null-value1 = NUMERIC-NULL

IF ISQ-TYPER(SBL-ISQ-PTR(SBL-INDEX)) = "C"

   null-value2 = CHARACTER-NULL

ELSE

   null-value2 = NUMERIC-NULL
```

3. Exit processing.

22.20.5 Outputs

1. Status of function processing.

```
RET-STATUS   PIC X(5).
```

2. Generated IF Statement on the specified file.

22.21 Function CDRPUIF - Generate COBOL IF Statement for Union Discriminator for Specified Record Type

This function will generate a COBOL IF statement that will evaluate record union discriminator qualification for a specified record type. The IF statement will be generated into a request processor sub- program to perform the evaluation at the internal schema level.

22.21.1 Inputs

1. Boolean Operators, conditions and parenthesis from the NDML WHERE clause.

```
SUBTRANS-BOOLEAN-LIST
```

2. Internal Schema Representation of the NDML command.

```
IS-QUALIFY-LIST
```

3. Record Type Identification.

```
RECORD-TYPE-NUMBER   PIC 9(6)
```

4. Name of the file where the IF Statement is to be generated.

```
FILE-NAME              PIC X(30)
```

5. Subtransaction Identification.

```
SUBTRANS-ID     PIC 9(3)
```

6. Data Base user defined null values.

```
CHARACTER-NULL  PIC X(30)
NUMERIC-NULL    PIC X(30)
```

22.21.2 CDM Requirements

None

22.21.3 Internal Requirements

None

22.21.4 Processing

1. Generate the start of an IF statement in the specified file. Generate:

```
IF
```

2. Generate an IF statement to evaluate record union discriminators. Process the portion of the SUBTRANS-BOOLEAN-LIST for the specified subtransaction. Perform steps 2.1 - 2.5 for each entry in the SUBTRANS-BOOLEAN-LIST where

```
SBL-SUBTRANS(SBL-INDEX) = SUBTRANS-ID
SBL-TYPE    (SBL-INDEX) = "2U"
SBL-RTNO    (SBL-INDEX) = RECORD-TYPE-NO
```

2.1 Process an operator or parenthesis from the SUBTRANS-BOOLEAN-LIST. Generate into the specified file, if SBL-OP(SBL-INDEX) not equal space

```
oper/paren

where

oper/paren = SBL-OP (SBL-INDEX)

Continue processing at step 2.
```

2.2 Process an external qualification condition in SUBTRANS-BOOLEAN-LIST. Generate into the specified file, if SBL-ISQ-PTR not equal zero and ISQ-OP(SBL-ISQ-PTR(SBL-INDEX)) not = "NN" or "NL" and ISQ-TYPE(SBL-ISQ-PTR(SBL-INDEX)) = "2" and SOURCE-IS-EXTERNAL(SBL-ISQ-PTR(SBL-INDEX))

```
(D-dfno op ISQ-VAR-qq AND
       D-dfno NOT = null-value)

       where

       dfno = ISQ-DFNOL(SBL-ISQ-PTR(SBL-INDEX))
       op   = ISQ-OP(SBL-ISQ-PTR(SBL-INDEX))
       qq   = SBL-ISQ-PTR(SBL-INDEX)

       IF ISQ-TYPEL(SBL-ISQ-PTR(SBL-INDEX)) = "C"

                  null-value = CHARACTER-NULL

       ELSE

                  null-value = NUMERIC-NULL

       Continue processing at step 2.1
```

2.3 Process an IS NULL or IS NOT NULL externa. qualification condition in the SUBTRANS-BOOLEAN-LIST. Generate into the specified file, if SBL-ISQ-PTR not equal zero and ISQ-OP(SBL-ISQ-PTR(SBL-INDEX)) = "NN" or "NL" and ISQ-TYPE(SBL-ISQ-PTR(SBL-INDEX)) = "2" and SOURCE-IS-EXTERNAL(SBL-ISQ-PTR(SBL-INDEX))

```
   D-dfno op null-value

where

   dfno = ISQ-DFNOL(SBL-ISQ-PTR(SBL-INDEX))

IF ISQ-TYPEL(SBL-ISQ-PTR(SBL-INDEX)) = "C"

   null-value = CHARACTER-NULL

ELSE

   null-value = NUMERIC-NULL

IF ISQ-OP(SBL-ISQ-PTR(SBL-INDEX)) = "NN"

   op = "="

ELSE

   op = "NOT ="

Continue processing at step 2.
```

2.4 Process a union qualification in SUBTRANS-BOOLEAN-LIST. Generate into the specified file if:

```
   SBL-ISQ-PTR(SBL-INDEX) NOT = ZERO AND
   ISQ-TYPE(SBL-ISQ-PTR(SBL-INDEX)) = "2" AND
   SOURCE-IS-UNION(SBL-ISQ-PTR(SBL-INDEX))
```

```
(D-dfno op value AND
 D-dfno NOT = null-value)

where

   dfno = ISQ-DFNOL(SBL-ISQ-PTR(SBL-INDEX))
   op   = ISQ-OP(SBL-ISQ-PTR(SBL-INDEX))

If ISQ-TYPEL(SBL-ISQ-PTR(SBL-INDEX)) = "C"

   value = "ISQ-UNION-VALUE(SBL-ISQ-PTR(SBL-INDEX))"
   null-value = CHARACTER-NULL

ELSE

   value = ISQ-UNION-VALUE(SBL-ISQ-PTR(SBL-INDEX))
   null-value = NUMERIC-NULL

Continue processing at step 2.1.
```

2.5 Process an intra-subtransaction qualification in SUBTRANS-BOOLEAN-LIST, generate into the specified file if:

```
   SBL-ISQ-PTR(SBL-INDEX) NOT = ZERO AND
   ISQ-TYPE(SBL-ISQ-PTR(SBL-INDEX)) = "3"

   (D-dfno1 op D-dfno2 AND
    D-dfno1 NOT = null-value1 AND
    D-dfno2 NOT = null-value2)

where

   dfno1 = ISQ-DFNOL(SBL-ISQ-PTR(SBL-INDEX))
   op    = ISQ-OP(SBL-ISQ-PTR(SBL-INDEX))
   dfno2 = ISQ-DFNOR(SBL-ISQ-PTR(SBL-INDEX))

IF ISQ-TYPEL(SBL-ISQ-PTR(SBL-INDEX)) = "C"

   null-value1 = CHARACTER-NULL

ELSE

   null-value1 = NUMERIC-NULL

IF ISQ-TYPER(SBL-ISQ-PTR(SBL-INDEX)) = "C"

   null-value2 = CHARACTER-NULL

ELSE

  null-value2 = NUMERIC-NULL
```

3. Exit processing.

22.21.5 Outputs

1. Status of function processing.

```
   RET-STATUS            PIC X(5).
```

2. Generated IF Statement on the specified file.

SECTION 23

FUNCTION PRE9.2 GENERATE SQL REQUEST PROCESSOR

The PRE9 SQL Request Processor (CDQPS) will generate the COBOL code required to execute NDML subtransactions against a Relational database. Currently ORACLE and DB2 are supported.

The generated program will use ORACLE as a Relational database management system to control all the transactions against the database. The program will support both update and retrieval requests against the database. The results of all retrieval operations will be reformatted into a sequential file and returned to the requesting application.

The DB2 VARCHAR datatype is not supported.

23.1 Inputs:

Inputs are the outputs from the following functions:

PRE5 - Decompose CS NDML
PRE6 - Select IS Access Path
PRE7 - Transform IS Access Path/Generic DML

1. RPS-DBMS PIC X(30)

   RPS-DBMS will contain the name of the target database management system. Currently, only ORACLE and DB2 are supported.

2. MY-HOST PIC XXX

   MY-HOST will contain the name of the host upon which this function will execute.

3. RPS-RPID PIC X(10)

   RPS-RPID will contain the PROGRAM-ID of the request processor to be generated.

4. RPS-SUBTRANS PIC 999

   RPS-SUBTRANS will contain the subtransaction number.

5. IS-ACTION-LIST in ISAL copy member of IISSCLIB

   IS-ACTION-LIST will contain the internal representation of the retrieval or update fields.

6. IS-QUALIFY-LIST in ISQUAL copy member of IISSCLIB

   IS-QUALIFY-LIST will contain the internal representation of the WHERE clause.

7. CS-ACTION-LIST in CSAL copy member of IISSCLIB

   CS-ACTION-LIST will contain the conceptual representation of the retrieval or update fields.

8. CS-QUALIFY-LIST in CSQUAL copy member of IISSCLIB

   CS-QUALIFY-LIST will contain the conceptual representation of the WHERE clause.

9. RFT in RFTABLE copy member of IISSCLIB

   RFT will contain the conceptual descriptions of requested data fields.

10. SUBTRANS-BOOLEAN-LIST in SUBBOOL copy member of IISSCLIB

    SUBTRANS-BOOLEAN-LIST will contain evaluatable search criteria ordered by SUBTRANS-ID.

11. CMA-TABLE in copy member of IISSCLIB

    CMA-TABLE contains the information required to generate calls to complex mapping algorithms.

12. CHAR-NULL PIC X(30)

    CHAR-NULL contains the keyword NULL, indicating that the default database null value is to be used or the user specified null value for non-numeric fields.

13. NUMERIC-NULL PIC X(30)

    NUMERIC-NULL contains the keyword NULL, indicating that the default database null value is to be used or the user specified null value for numeric fields.

14. DB-USER-ID PIC X(30)

    DB-USER-ID contains the user identification under which the request processor will execute.

15. BOOLEAN-LIST

    BOOLEAN-LIST contains parenthesized logic and boolean operators for the WHERE clause.

23.2 CDM Requirements

None

23.3 Internal Requirements

None

Macro Generation:

Macros are code templates with optional substitutable parameters which allow generated code to be more independent of the generating programs. All macros are to be generated through calls to CDMACR. This routine requires the following parameters:

```
INPUT
 FILE-NAME     PIC X(30)     included in MACDAT copy member
 LIBRARY-NAME  PIC X(30)     included in MACDAT copy member
 MACRO-NAME    PIC X(8)      included in MACDAT copy member
 SUBSTITUTION-LIST           included in SBSTLST copy member

OUTPUT
 RET-STATUS    PIC X(5)
```

FILE-NAME contains the name of the file to which code is generated. This file must be closed prior to the CDMACR call. Upon return to CDQPS, FILE-NAME must be reopened for EXTEND to allow code to be generated at the end of the file.

LIBRARY NAME contains an identifier common to macros supporting similar functions. All CDQPS macros will have LIBRARY-NAME equal SQL.

MACRO-NAME contains the name of the macro to be generated, for example CDQPS01.

SUBSTITUTION-LIST is described by the following structure:

```
01 SUBSTITUTION-LIST.
   03 SL-USED      PIC 99.
   03 SL-MAX       PIC 99.
   03 SL-ROW-SIZE PIC 99.
   03 SL-ENTRY OCCURS 8 TIMES
      INDEXED BY SL-INDEX.
   05 SL-PARAMETER PIC X(30)
   05 SL-SUBST-VAL PIC X(30)
```

SUBSTITUTION-LIST is populated by setting SL-USED to the number of parameter values the macro requires. SL-PARAMETER (index) contains the macro parameter to be substituted, for example P1. SL-SUBST-VAL (INDEX) contains the corresponding substitution value, for example, IS-DFNO.

23.4 Processing

1. Generate a file, WORK-FILE, to contain generated code using GENFIL. Open the file for OUTPUT. Move the file name to GEN-FILE-NAME. Call GENFIL using:

```
INPUT
   MY-HOST      XXX
   WORK-FILE    X(30)
OUTPUT
   RET-STATUS   X(5)
```

2. Build the identification division.

   If a select, a type 1 referential integrity test, a type 2 referential integrity test, a query combination command, or a key uniqueness test request processor is to be generated, (IS-ACTION equals S or 1 or 2 or Q or K), generate macro CDQPS01 into WORK-FILE substituting the contents of CDQPS input parameter RPS-RPID for parameter P1.

   If a modify, delete or insert request processor is to be generated, (IS-ACTION equals M or D or I), generate macro CDQPS02 into WORK-FILE. Substitute the value of RPS-RPID for P1. If RPS-DBMS equals ORACLE, substitute the value 1403 for P2. If RPS-DBMS equals DB2, substitute the value 100 for P2.

3. If generating a Select, a type 1 referential integrity test, a type 2 referential integrity test, a query combination command, or a key uniqueness test, call CDRFT to generate the results record, null flag and field definitions. In addition, CDRFT generates the working storage section header and conceptual schema data field definitions.

   To call CDRFT, close WORK-FILE and pass the following parameters. Upon return, reopen WORK-FILE for EXTEND.

```
INPUT
  WORK-FILE     PIC X(30)
  RPS-SUBTRANS  PIC 999
  RFT
OUTPUT
  NO PARAMETERS
```

4. If generating a Select, type 1 referential integrity test, a type 2 referential integrity test, a query combination command, or a key uniqueness test, generate in WORK-FILE working storage variables using the CDQPS03 macro. Substitute the value of RPS-RPID for P1. If RP-DBMS equals ORACLE, substitute 1403 for P2. If RP-DBMS equals DB2, substitute 100 for P2.

5. Generate, in WORK-FILE, the following declaration header, placing an * in column 7 if RPS-DBMS is DB2.

```
XEC SQL BEGIN DECLARE SECTION END-EXEC.
```

6. Generate, in WORK-FILE, the data elements for the search/update values by closing WORK-FILE, calling CDPRM and reopening the work file. Send CDPRM the following parameters.

```
INPUT
  WORK-FILE     PIC X(30)
  RPS-SUBTRANS  PIC 999
  IS-QUALIFY-LIST
  COMPLEX-MAPPING-ALG-TABLE
OUTPUT
  NO PARAMETERS
```

7. Generate, in WORK-FILE, the update data fields and associated indicator variables by closing WORK-FILE, calling CDRDF with the following parameters and reopening WORK-FILE for EXTEND upon return.

```
INPUT
  WORK-FILE     PIC X(30)
  RPS-SUBTRANS  PIC 999
  IS-ACTION-LIST
  COMPLEX-MAPPING-ALG-TABLE
OUTPUT
  NO PARAMETERS
```

8. Generate, in WORK-FILE, the retrieval data fields and associated indicator variables by closing WORK-FILE, calling CDGDF with the following parameters and reopening WORK-FILE for EXTEND upon return.

```
INPUT
  IS-ACTION-LIST
  IS-QUALIFY-LIST
  COMPLEX-MAPPING-ALG-TABLE
  RPS-SUBTRANS   PIC 9(3)
  WORK-FILE      PIC X(30)
OUTPUT
  RET-STATUS     PIC X(30)
```

9. Generate the working storage variable definitions for the user defined null values which will be used in qualification at the data base level by closing the file, calling CDGNV with the following parameters and reopening the file for extend upon return.

```
INPUT
  IS-QUALIFY-LIST
  RPS-SUBTRANS    PIC 9(3)
  NUMERIC-NULL    PIC X(30)
  CHAR-NULL       PIC X(30)
```

```
    WORK-FILE       PIC X(30)
OUTPUT
    NO PARAMETERS
```

10. Generate, in WORK-FILE, the declaration trailer, placing an * in column 7 if RPS-DBMS is DB2.

```
EXEC SQL END DECLARE SECTION END-EXEC.
```

11. Generate on WORK-FILE, if necessary, record and datafield definitions in support of tag(s) to record type and record type to tag complex mappings.

    Scan the CMA-TABLE looking for all used CMA-ALG-ENTRYs whose CMA-SUBTRANSACTION equals RPS-SUBTRANS. For each qualifying CMA-ALG-ENTRY, search for all CMA-PARM-ENTRYs which have a non-zero CMA-RT-NO and a zero CMA-DF-NO. For each unique CMA-RT-NO, perform the following steps after closing WORK-FILE.

    11.1 Generate the following line:

```
01 T-rtno.
```

    where rtno is the value of the current CMA-RT-NO.

    11.2 Call CDGENRT with the following parameters

```
INPUTS
  CMA-DB-ID     PIC 9(6)
  CMA-RTID      PIC X(30)
  WORK-FILE     PIC X(30)
OUTPUTS
  RET-STATUS    PIC X(5)
```

    11.3 Upon return from each call, check the status parameter and generate error messages as necessary. Upon return from the final call, reopen WORK-FILE for EXTEND.

12. Generate on WORK-FILE, if necessary, the complex mapping parameter definitions by closing WORK-FILE, calling CDCMPRM with the following parameters and reopening WORK-FILE for EXTEND upon return.

```
INPUT
  COMPLEX-MAPPING-ALG-TABLE
  RPS-SUBTRANS PIC 999
```

```
  WORK-FILE     PIC X(30)
OUTPUT
  RET-STATUS    PIC X(5)
```

Check the status parameter and generate appropriate error messages as necessary.

13. If IS-ACTION equals D or M, call CDGTV after closing the file to generate tag number variables and indicators to support complex mapping in the qualify list. Reopen the file for EXTEND upon return.

```
INPUT
 COMPLEX-MAPPING-ALG-TABLE
 IS-QUALIFY-LIST
 CS-QUALIFY-LIST
 WORK-FILE      PIC X(30)
 RPS-SUBTRANS   PIC 9(3)
OUTPUT
 RET-STATUS     PIC X(5)
```

14. Generate the INCLUDE statement for the SQL status variables.

```
EXEC SQL INCLUDE SQLCA END-EXEC.
```

15. Generate the LINKAGE SECTION header and the MESSAGE-BODY-IN portion of the linkage section by closing WORK-FILE, calling CDMSG with the following parameters and reopening WORK-FILE for EXTEND upon return.

```
INPUT
  WORK-FILE     PIC X(30)
  RPS-SUBTRANS PIC 999
  CS-QUALIFY-LIST
  CS-ACTION-LIST
  IS-QUALIFY-LIST
  IS-ACTION-LIST
OUTPUT
  NO PARAMETERS
```

16. Generate, on WORK-FILE, the second half of the linkage section and the procedure division initialization logic by generating macro CDQPS04 which has no parameters.

17. If processing a select, type 1 referential integrity test, type 2 referential integrity test, query combination command, or key uniqueness test (IS-ACTION

equals S or 1 or 2 or K) generate on WORK-FILE the results filename generation logic by generating macro CDQPS05 which has no parameters.

18. Populate the user defined character null value variables, if they exist, by generating move statements for each unique internally evaluatable character length represented in the IS-QUALIFY list if a user defined null character value exists.
If CHAR-NULL doesn't equal "NULL", scan IS-QUALIFY-LIST looking for all ISQ-SUBTRANS-ID equal to RPS-SUBTRANS and ISQ-EVAL-FLAG > 0. For each qualifying entry, check to see if ISQ-TYPEL = "C" or ISQ-TYPER = "C. For each qualifying type, generate the appropriate MOVE statement.

```
MOVE charnull TO DB-CHAR-NULL-nn.
```

where charnull is the CHAR-NULL value and nn is the ISQ-SIZEL or ISQ-SIZER value.

19. Populate the user defined numeric null variable, if it exists, by generating a move statement if an internally evaluatable numeric entry appears in the IS-QUALIFY-LIST and if a user defined numeric null value exists. If NUMERIC-NULL doesn't equal "NULL", generate the MOVE statement.

```
MOVE numnull TO DB-NUM-NULL.
```

where numnull is the NUMERIC-NULL value.

20. Generate, in WORK-FILE, the conceptual to internal schema transform logic by closing WORK-FILE, calling CDCI with the following parameters and reopening WORK-FILE for EXTEND upon return.

```
INPUT
  WORK-FILE      PIC X(30)
  RPS-SUBTRANS   PIC 999
  IS-ACTION-LIST
  IS-QUALIFY-LIST
  COMPLEX-MAPPING-ALG-TABLE
  NUMERIC-NULL   PIC X(30)
  CHAR-NULL      PIC X(30)
OUTPUT
  RET-STATUS     PIC X(5)
```

Check the status parameter and generate error messages as necessary.

21. If processing a select, type 1 referential integrity test, type 2 referential integrity test or key uniqueness test (IS-ACTION equals S or 1 or 2 or K), perform the following steps, otherwise go to step 22.

21.1 Generate the DECLARE CURSOR statement.

21.1.1 Generate the first line.

EXEC SQL DECLARE IISSCUR CURSOR FOR

21.1.2 Generate the SELECT keyword.

SELECT

21.1.3 Generate the column list.

Scan the IS-ACTION-LIST searching for all entries which have IS-SUBTRANS-ID equal RPS-SUBTRANS, both IS-DFNO and IS-RTNO not equal zero, IS-DELETE-FLAG not equal 1, and IS-ALG-PTR = 0.

Scan the COMPLEX-MAPPING-ALG-TABLE searching for all entries which have CMA-SUBTRANSACTION equal RPS-SUBTRANS. For each subtrans satisfying the argument, search for non-zero CMA-RT-NO. If found, look for non-zero CMA-DF-NO. If not found, scan CMA-DF-ENTRY searching for DF-MOD-PTR equal CMA-INDEX and DF-PARM-PTR equal CMA-PARM-INDEX.

For each entry found, generate 1 line according to one of the following formats, separating each line from the next by a comma except for the last line.

If generating ORACLE, generate

rtid1.dfid1,
.
.
.

rtidn.dfidn

If generating DB2, generate

user.rtid1.dfid1,
.
.
.
user.rtidn.dfidn

where user is the DB-USER-ID value, rtid is the IS-RTID value and dfid is the IS-DFID value.

21.1.4 Generate the FROM keyword.

FROM

21.1.5 Generate the FROM table list.

Search the IS-ACTION-LIST for each unique IS-RTID, which has IS-SUBTRANS-ID equal RPS-SUBTRANS and IS-DELETE-FLAG not equal 1. Also search the IS-QUALIFY-LIST for all not previously generated unique ISQ-RTIDL'S and non-blank ISQ-RTIDR'S which have their corresonding ISQ-SUBTRANS-IDL'S and ISQ-SUBTRANS-IDR'S equal RPS-SUBTRANS. For each occurrence, generate additional FROM table list entries. Generate 1 line according to one of the following formats, separating each line from the next by a comma except for the last line.

For ORACLE, generate

rtid1,
.
.
.
rtidn

For DB2, generate

user.rtid1,
.
.
.
user.rtidn

where user is the DB-USER-ID value and rtid is the IS-RTID value.

21.1.6 Generate the WHERE clause.

21.1.6.1 Search the SUBTRANS-BOOLEAN-LIST looking for all used SBL-ENTRYs with SBL-SUBTRANS equal RPS-SUBTRANS.

If no matching entries are found, no WHERE clause is to be generated. Go to step 21.1.7 or, if this was performed in another section, go to the step following the WHERE clause generation.

If at least 1 matching entry is found, generate the WHERE keyword.

WHERE

21.1.6.2 Perform the following steps for each SBL-ENTRY with SBL-SUBTRANS equal RPS-SUBTRANS. After all SBL-ENTRYs have been processed, go to step 21.1.7 or, if this was performed in another section, go to the step following the WHERE clause generation.

21.1.6.2.1 If the current SBL-ISQ-PTR equals zero, generate the SBL-OP.

sblop

where sblop is the value of the current SBL-OP.

Return to step 21.1.6.2 to process the next matching SBL-ENTRY.

21.1.6.2.2 Generate an open parenthesis.

(

21.1.6.2.3 If the ISQ-TYPE equals 2 and the ISQ-TYPE2-SOURCE not equal U on the ISQ entry pointed to by SBL-ISQ-PTR, perform the following steps, otherwise go to step 21.1.6.2.4.

21.1.6.2.3.1 Generate the following line.

If ORACLE, generate

rtidl.dfidl

If DB2, generate

user.rtidl.dfidl

where user is the DB-USER-ID value, rtidl is the current ISQ-RTIDL value and dfidl is the current ISQ-DFIDL value.

21.1.6.2.3.2 If the current ISQ-OP equal NL, generate the following, otherwise go to step 21.1.6.2.3.3.

IS NULL

If the current ISQ-TYPEL equal C and CHAR-NULL not equal NULL, generate

OR (on 1 line)

and if ORACLE, on the next line

rtidl.dfidl = :DB-CHAR-NULL-isqsizel

and if DB2, on the next line

user.rtidl.dfidl = :DB-CHAR-NULL-isqsizel

where user is the DB-USER-ID value, rtidl is the ISQ-RTIDL value, dfidl is the ISQ-DFIDL value and isqsizel is the ISQ-SIZEL value.

If the current ISQ-TYPEL not equal C and NUMERIC-NULL not equal NULL, generate

OR (on 1 line)

and if ORACLE, on the next line

rtidl.dfidl = :DB-NUM-NULL

and if DB2, on the next line

user.rtidl.dfidl = :DB-NUM-NULL

where user is the DB-USER-ID value, rtidl is the ISQ-RTIDL value and dfidl is the ISQ-DFIDL value.

Generate a close parenthesis.

)

Return to step 21.1.6.2 to process the next matching SBL-ENTRY.

21.1.6.2.3.3 If the current ISQ-OP equal NN, generate the following, otherwise go to step 21.1.6.2.3.4.

IS NOT NULL

If the current ISQ-TYPEL equal C and CHAR-NULL not equal NULL, generate

AND (on 1 line)

and if ORACLE, on the next line

rtidl.dfidl != :DB-CHAR-NULL-isqsizel

and if DB2, on the next line

user.rtidl.dfidl ^=
:DB-CHAR-NULL-isqsizel

where user is the DB-USER-ID value, rtidl is the ISQ-RTIDL value, dfidl is the ISQ-DFIDL value and isqsizel is the ISQ-SIZEL value.

If the current ISQ-TYPEL not equal C and NUMERIC-NULL not equal null, generate

AND (on 1 line)

and if ORACLE, on the next line

rtidl.dfidl != :DB-NUM-NULL

and if DB2, on the next line

user.rtidl.dfidl ^= :DB-NUM-NULL

where user is the DB-USER-ID value, rtidl is the ISQ-RTIDL value and dfidl is the ISQ-DFIDL value.

Generate a close parenthesis.

)

Return to step 21.1.6.2 to process the next matching SBL-ENTRY.

21.1.6.2.3.4 If ISQ-OP equal != and DB2, generate

^=

Otherwise, generate the ISQ-OP.

op

where op is the current ISQ-OP value.

21.1.6.2.3.5 Generate the ISQ variable.

:ISQL-VAR-isqindex

where isqindex is the current ISQ-INDEX.

21.1.6.2.3.6 If the current ISQ-TYPEL equal C and CHAR-NULL not equal NULL, generate

AND (on 1 line)

and if ORACLE, on the next line

rtidl.dfidl != :DB-CHAR-NULL-isqsizel

and if DB2, on the next line

user.rtidl.dfidl ^=
:DB-CHAR-NULL-isqsizel.

where user is the DB-USER-ID value, rtidl is the ISQ-RTIDL value, dfidl is the ISQ-DFIDL value and isqsizel is the ISQ-SIZEL value.

If the current ISQ-TYPEL not equal C and NUMERIC-NULL not equal NULL, generate

AND (on 1 line)

and if ORACLE, on the next line

rtidl.dfidl != :DB-NUM-NULL

and if DB2, on the next line

user.rtidl.dfidl ^= :DB-NUM-NULL

where user is the DB-USER-ID value, rtidl is the ISQ-RTIDL value and dfidl is the ISQ-DFIDL value.

Generate a close parenthesis.

)

Return to step 21.1.6.2 to process the next matching SBL-ENTRY.

21.1.6.2.4 If the ISQ-TYPE equals 2 and ISQ-TYPE2-SOURCE equals U (union discriminator) in the ISQ entry pointed to by SBL-ISQ-PTR, perform the following steps, otherwise go to step 21.1.6.2.5.

21.1.6.2.4.1 Generate the following line.

If ORACLE,

rtidl.dfidl

If DB2,

user.rtidl.dfidl

where user is the DB-USER-ID value, rtidl is the ISQ-RTIDL value and dfidl is the ISQ-DFIDL value.

21.1.6.2.4.2 If ISQ-OP equal != and DB2, generate

^=

Otherwise, generate the ISQ-OP

op

where op is the ISQ-OP value.

21.1.6.2.4.3 If ISQ-TYPEL equal C, generate

"unionval"

where unionval is the ISQ-UNION-VALUE value. Place quotes around the value so as to make the character string length equal ISQ-SIZEL.

If ISQ-TYPEL not equal C, generate

unionval

where unionval is the ISQ-UNION-VALUE value.

Generate a close parenthesis.

)

Return to step 21.1.6.2 to process the next matching SBL-ENTRY.

21.1.6.2.5 If the ISQ-TYPE equal 3 in the ISQ entry pointed to by SBL-ISQ-PTR, perform the following steps.

21.1.6.2.5.1 Generate the following line.

If ORACLE,

rtidl.dfidl

If DB2,

user.rtidl.dfidl

where user is the DB-USER-ID value, rtidl is the ISQ-RTIDL value and dfidl is the ISQ-DFIDL value.

21.1.6.2.5.2 If ISQ-OP equal != and DB2, generate

^=

Otherwise, generate the ISQ-OP

op

where op is the ISQ-OP value.

21.1.6.2.5.3 Generate the following line.

If ORACLE,

rtidr.dfidr

If DB2,

user.rtidr.dfidr

where user is the DB-USER-ID value, rtidr is the ISQ-RTIDR value and dfidr is the ISQ-DFIDR value.

21.1.6.2.5.4 If ISQ-TYPEL equal C and CHAR-NULL not equal NULL, generate

AND (on 1 line)

and if ORACLE, on the next line

rtidl.dfidl != :DB-CHAR-NULL-isqsizel

and if DB2, on the next line

user.rtidl.dfidl ^= : DB-CHAR-NULL-isqsizel

where user is the DB-USER-ID value, rtidl is the ISQ-RTIDL value, dfidl is the ISQ-DFIDL value and isqsizel is the ISQ-SIZEL value.

If the current ISQ-TYPEL does not equal C and NUMERIC-NULL does not equal NULL, generate

AND (on 1 line)

and if ORACLE, on the next line

rtidl.dfidl != :DB-NUM-NULL

and if DB2, on the next line

user.rtidl.dfidl ^= :DB-NUM-NULL

where user is the DB-USER-ID value, rtidl is the ISQ-RTIDL value and dfidl is the ISQ-DFIDL value.

21.1.6.2.5.5 If ISQ-TYPER equal C and CHAR-NULL not equal NULL, generate

AND (on 1 line)

and if ORACLE, on the next line

rtidr.dfidr != :DB-CHAR-NULL-isqsizer

and if DB2, on the next line

user.rtidr.dfidr ^= :DB-CHAR-NULL-isqsizer

where user is the DB-USER-ID value, rtidr is the ISQ-RTIDR value, dfidr is the ISQ-DFIDR value and isqsizer is the ISQ-SIZER value.

If ISQ-SIZER not equal C and NUMERIC-NULL not equal NULL, generate

AND (on 1 line)

and if ORACLE, on the next line

rtidr.dfidr != :DB-NUM-NULL

and if DB2, on the next line

user.rtidr.dfidr ^= :DB-NUM-NULL

where user is the DB-USER-ID value, rtidr is the ISQ-RTIDR value, and dfidr is the ISQ-DFIDR value.

Generate a close parenthesis.

)

Return to step 21.1.6.2 to process the next matching SBL-ENTRY.

21.1.7 Generate the following line.

END-EXEC.

21.2 Generate the OPEN CURSOR statement.

EXEC SQL OPEN IISSCUR END-EXEC.

21.3 Generate the error checking logic by substituting KES-SELECT-ERROR for P1 and UNABLE TO OPEN CURSOR for file P2 in macro CDQPS06.

21.4 Generate the beginning of the FETCH logic by generating macro CDQPS07 which has no parameters.

21.5 Initialize all receiving fields. Scan the IS-ACTION-LIST for all unique, non-zero IS-DFNO/IS-RTNO combinations having IS-SUBTRANS-ID equal RPS-SUBTRANS, IS-ALG-PTR = 0, and IS-DELETE-FLAG not equal 1. Scan the COMPLEX-MAPPING-ALG-TABLE searching for all entries which have CMA-SUBTRANSACTION equal RPS-SUBTRANS. For each SUBTRANS satisfying the argument, search for non-zero CMA-RT-NO. If found, look for non-zero CMA-DF-NO. If not found, scan CMA-DF-ENTRY searching for DF-MOD-PTR equal CMA-INDEX and DF-PARM-PTR equal CMA-PARM-INDEX. Generate the following two lines for each entry found.

```
MOVE ZERO TO D-dfno-rtno.
MOVE ZERO TO INDP-dfno-rtno.
```

where dfno is the IS-DFNO value and rtno is the IS-RTNO value.

21.6 Generate the FETCH statement.

21.6.1 Generate the following line.

```
EXEC SQL FETCH IISSCUR INTO
```

21.6.2 For each entry selected in step 21.5, generate the following line, separating each line from the next by a comma except for the last line.

```
:D-dfno-rtno:INDP-dfno-rtno
```

where dfno is the IS-DFNO value and rtno is the IS-RTNO value.

21.6.3 Generate the following line.

```
END-EXEC.
```

21.7 Generate FETCH error handling logic by substituting ORACLE or DB2 depending on the target database for P1 in macro CDQPS08.

21.8 Generate the internal to conceptual transform logic by closing the work file, calling CDIC with the following parameters and reopening the work file for EXTEND upon return.

```
INPUTS
  WORK-FILE                PIC X(30)
  RPS-SUBTRANS                      PIC 9(3)
  IS-ACTION-LIST
  COMPLEX-MAPPING-ALG-TABLE
  NUMERIC-NULL                      PIC X(30)
  CHAR-NULL                         PIC X(30)
  CS-QUALIFY-LIST
  IS-QUALIFY-LIST
OUTPUT
  RET-STATUS                        PIC X(5)
```

21.9 Generate the end of the fetch loop and the program end by generating macro CDQPS09 which has no parameters.

21.10 Go to step 25.

22. Generate the remainder of the subroutine for inserts.

If IS-ACTION equal I, perform the following steps, otherwise go to step 23.

22.1 Generate the first line of the insert.

EXEC SQL INSERT INTO

22.2 Generate the tablename.

If ORACLE, generate

rtid

If DB2, generate

user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

22.3 Generate an open parenthesis.

(

22.4 Scan the IS-ACTION-LIST for entry with IS-SUBTRANS-ID equal RPS-SUBTRANS and IS-DELETE-FLAG not equal 1. If IS-ALG-PTR not equal zero, set CMA-INDEX to IS-ALG-PTR. Scan the CMA-PARM-ENTRY for non-zero CMA-DF-NO. If IS-ALG-PTR equals zero, use the IS-DFID. For each entry found, generate the following line, separating each line from the next by a comma except for the last.

dfid

where dfid is the IS-DFID value or the CMA-DFID value.

22.5 Generate a close parenthesis.

)

22.6 Generate the VALUES keyword and an open parenthesis.

```
VALUES (
```

22.7 Generate the insert value and indicator variable names.

Scan the IS-ACTION-LIST. For each used entry with IS-SUBTRANS-ID equal RPS-SUBTRANS and IS-DELETE-FLAG not equal 1, test the IS-ALG-PTR. If the IS-ALG-PTR equals zero, generate

```
:IS-VAR-isindex:INDP-isindex
```

where isindex is the IS-INDEX value.

If the IS-ALG-PTR does not equal zero, set CMA-INDEX to IS-ALG-PTR. Scan the CMA-PARM-ENTRY for non-zero CMA-DFNO. For each entry found, generate

```
:IS-mod-inst-pno:INDP-mod-inst-pno
```

where mod is the IS-ALG-ID value, inst is the CMA-MOD-INST value pointed to by the IS-ALG-PTR and pno is the CMA-PARM-NO value with non-zero CMA-DFID.

Separate each line from the next by commas except for the last.

22.8 Generate the following lines.

```
)

END-EXEC.
```

22.9 Generate the error handling logic by substituting, if ORACLE, KES-ORACLE-INSERT-ERROR and if DB2, KES-DB2-INSERT-ERROR for P1 and UNABLE TO INSERT for P2 in macro CDQPS06.

22.10 Generate the end program logic by generating macro CDQPS10 which has no parameters.

22.11 Go to step 25.

23. Generate the remainder of the subroutine for modifys.

If IS-ACTION equals M, perform the following steps, otherwise go to step 24.

23.1 If all used ISQ-RTIDLs and non-blank ISQ-RTIDRs equal IS-RTID (1) (no USING clause) and all used ISQ-EVAL-FLAGs do not equal zero (everything is evaluatable internally), perform the following steps, otherwise go to step 23.2.

23.1.1 Generate the EXEC SQL statement.

EXEC SQL

23.1.2 Generate the UPDATE statement.

If ORACLE, generate

UPDATE rtid

If DB2, generate

UPDATE user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

23.1.3 Generate the SET keyword.

SET

23.1.4 Generate the column = variable portion of the UPDATE statement.

Scan the IS-ACTION-LIST. For each used entry with IS-SUBTRANS-ID equals RPS-SUBTRANS and IS-DELETE-FLAG not equal 1, test the IS-ALG-PTR. If the IS-ALG-PTR equals zero, generate

dfid

On the next line, generate

=

On the next line, generate

:IS-VAR-isindex:INDP-isindex

where dfid is the IS-DFID value and isindex is the IS-INDEX value.

If the IS-ALG-PTR does not equal zero, set CMA-INDEX to IS-ALG-PTR. Scan the CMA-PARM-ENTRY searching for CMA-DFID not equal spaces. Generate

dfid

On the next line, generate

=

On the next line, generate

:IS-mod-inst-pno:INDP-mod-inst-pno

where dfid is the IS-DFID value, mod is the IS-ALG-ID value, inst is the CMA-MOD-INST value pointed to by IS-ALG-PTR and pno is the IS-PARM-NO value with non-blank CMA-DFID.

Separate each 3 line entry from the next by a comma except for the last.

23.1.5 Perform steps 21.1.6 through 21.1.6.2.5.5 to generate the WHERE clause.

23.1.6 Generate the END-EXEC statement.

END-EXEC.

23.1.7 Generate the error checking logic by substituting, if ORACLE, KES-ORACLE-UPDATE-ERROR and if DB2, KES-DB2-UPDATE-ERROR for P1 and UNABLE TO UPDATE for P2 in macro CDQPS06A.

23.1.8 Generate the program end by generating macro CDQPS10 which has no parameters.

23.1.9 Go to step 25.

23.2 Generate the table lock statement.

EXEC SQL LOCK TABLE

If ORACLE, generate

rtid

If DB2, generate

user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

Generate

IN EXCLUSIVE MODE END-EXEC.

23.3 Generate status checking logic by substituting either ORACLE or DB2, depending on the target database for P1 and IS-RTID (1) for P2 in macro CDQPS11.

23.4 If at least 1 used ISQ-RTIDL or non-blank ISQ-RTIDR does not equal IS-RTID (1), a using clause has been employed. Perform the following steps, otherwise go to step 23.5.

23.4.1 Generate the DECLARE CURSOR statement.

23.4.1.1 Generate the first line.

EXEC SQL DECLARE IISSCUR CURSOR FOR

23.4.1.2 Generate the SELECT keyword.

SELECT

23.4.1.3 Generate the column list.

Scan the IS-ACTION-LIST searching for all entries which have IS-SUBTRANS-ID equal RPS-SUBTRANS, IS-DELETE-FLAG not equal 1, and which have both IS-DFNO and IS-RTNO not equal zero.

For each IS entry found, generate 1 line according to one of the following formats, separating each line from the next by a comma except for the last line.

If ORACLE, generate

```
rtid1.dfid1,
      .
      .
      .
rtidn.dfidn
```

If DB2, generate

```
user.rtid1.dfid1,
      .
      .
      .
user.rtidn.dfidn
```

where user is the DB-USER-ID value, rtid is the IS-RTID value and dfid is the IS-DFID value.

23.4.1.4 Generate the FROM keyword.

FROM

23.4.1.5 Generate the FROM table list.

For each unique IS-RTID, ISQ-RTIDL and ISQ-RTIDR which have their corresponding IS-SUBTRANS-ID, ISQ-SUBTRANS-IDL or ISQ-SUBTRANS-IDR equal RPS-SUBTRANS and, for the IS qualify entries, the ISQ-EVAL-FLAG not equal zero and the IS action entries with their IS-DELETE-FLAG not equal 1, generate 1 line according to one of the following formats, separating each line from the next by a comma except for the last line.

For ORACLE, generate

```
rtid1,
  .
  .
  .
rtidn
```

For DB2, generate

```
user.rtid1,
     .
     .
     .
user.rtidn
```

where user is the DB-USER-ID value and rtid is the IS-RTID, ISQ-RTIDL or ISQ-RTIDR value.

23.4.1.6 Perform steps 21.1.6 through 21.1.6.2.5.5 to generate the WHERE clause.

23.4.1.7 If ORACLE, generate the following line.

FOR UPDATE

23.4.1.8 If DB2, generate the following lines.

FOR UPDATE OF

Scan the IS-ACTION-LIST. For each entry which has both IS-DFNO and IS-RTNO not equal zero, IS-SUBTRANS-ID equal RPS-SUBTRANS, and IS-DELETE-FLAG not equal 1, generate

user.rtid.dfid

where user is the DB-USER-ID value, rtid is the IS-RTID value and dfid is the IS-DFID value.

Separate each line from the next by a comma except for the last line.

23.4.1.9 Generate the END EXEC statement.

END-EXEC.

23.4.2 Generate the OPEN CURSOR statement.

EXEC SQL OPEN IISSCUR END-EXEC.

23.4.3 Generate the error checking logic by substituting, if ORACLE, KES-ORACLE-OPEN-ERROR for P1 and if DB2, KES-DB2-OPEN-ERROR for P1 and UNABLE TO OPEN CURSOR for P2 in macro CDQPS06.

23.4.4 Generate the following label.

FETCH-LOOP.

23.4.5 Initialize all receiving fields. Scan the IS-ACTION-LIST. For each unique, non-zero IS-DFNO/IS-RTNO combination, with IS-SUBTRANS-ID equal RPS-SUBTRANS and IS-DELETE-FLAG not equal 1, generate the following two lines.

MOVE ZERO TO D-dfno-rtno.

MOVE ZERO TO INDP-dfno-rtno.

where dfno is the IS-DFNO value and rtno is the IS-RTNO value.

23.4.6 Generate the FETCH statement.

23.4.6.1 Generate the following line.

EXEC SQL FETCH IISSCUR INTO

23.4.6.2 For each entry selected in step 23.4.5, generate the following line, separating each line from the next by a comma except for the last line.

:D-dfno-rtno:INDP-dfno-rtno

where dfno is the IS-DFNO value and rtno is the IS-RTNO value.

23.4.6.3 Generate the following line.

END-EXEC.

23.4.7 Generate FETCH error handling logic by substituting ORACLE or DB2 depending on the target database for P1 in macro CDQPS08.

23.4.8 Generate the UPDATE statement.

23.4.8.1 Generate the following line.

EXEC SQL

23.4.8.2 If ORACLE, generate the following line.

UPDATE rtid

If DB2, generate

UPDATE user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

23.4.8.3 Generate the SET keyword.

SET

23.4.8.4 Generate the column = variable portion of the UPDATE statement.

Scan the IS-ACTION-LIST. For each used entry with IS-SUBTRANS-ID equal RPS-SUBTRANS and IS-DELETE-FLAG not equal 1, test the IS-ALG-PTR. If the IS-ALG-PTR equal zero, generate

dfid

On the next line, generate

=

On the next line, generate

:IS-VAR-isindex:INDP-isindex

where dfid is the IS-DFID value and isindex is the IS-INDEX value.

If the IS-ALG-PTR does not equal zero, set CMA-INDEX to IS-ALG-PTR. Scan the CMA-PARM-ENTRY searching for CMA-DFID not equal spaces. Generate

dfid

On the next line, generate

=

On the next line, generate

:IS-mod-inst-pno:INDP-mod-inst-pno

where dfid is the CMA-DFID value, mod is the IS-ALG-ID value, inst is the CMA-MOD-INST value pointed to by IS-ALG-PTR and pno is the IS-PARM-NO value with non-blank CMA-DFID.

Separate each 3 line entry from the next by a comma except for the last.

23.4.8.5 Generate the following two lines.

WHERE CURRENT OF IISSCUR
END-EXEC.

23.4.9 Generate the error checking logic by substituting, if ORACLE, KES-ORACLE-UPDATE-ERROR and if DB2, KES-DB2-UPDATE-ERROR for P1 and UNABLE TO UPDATE for P2 in macro CDQPS06.

23.4.10 Generate the following line.

GO TO FETCH-LOOP.

23.4.11 Generate the program end by generating macro CDQPS12 which has no parameters.

23.4.12 Go to step 25.

23.5 Generate the remainder of the program for the modify with no using clause and complex mapping in the qualification.

23.5.1 Generate the DECLARE CURSOR statement.

23.5.1.1 Generate the first line.

EXEC SQL DECLARE IISSCUR CURSOR FOR

23.5.1.2 Generate the SELECT keyword.

SELECT

23.5.1.3 Generate the column list.

Scan the IS-ACTION-LIST searching for all entries which have IS-SUBTRANS-ID equal RPS-SUBTRANS, IS-DELETE-FLAG not equal 1, IS-MAPPED-TO-FLAG = 'Y', and which have both IS-DFNO and IS-RTNO not equal zero.

In addition, scan the IS-QUALIFY list searching for non-zero ISQ-DFNOL/ISQ-RTNOL and ISQ-SUBTRANS-IDL equal RPS-SUBTRANS and ISQ-DFNOR/ISQ-RTNOR and ISQ-SUBTRANS-IDR equal RPS-SUBTRANS non-complex combinations.

Scan the COMPLEX-MAPPING-ALG-TABLE searching for all entries which have CMA-SUBTRANSACTION equal RPS-SUBTRANS. For each subtrans satisfying the argument, search for non-zero CMA-RT-NO. If found, look for non-zero CMA-DF-NO. If not found, scan CMA-DF-ENTRY searching for DF-MOD-PTR equal CMA-INDEX and DF-PARM-PTR equal CMA-PARM-INDEX.

For each IS-ACTION entry satisfying the above arguments, as well as each unique IS-QUALIFY entry or CMA entry, generate a SELECT list. Generate no duplicates.

If ORACLE, generate

rtid.dfid

If DB2, generate

user.rtid.dfid

where user is the DB-USER-ID value, rtid is either the IS-RTID, ISQ-RTIDL, ISQ-RTIDR or CMA-RTID value and dfid is either the IS-DFID, ISQ-DFIDL, ISQ-DFIDR, CMA-DF-ID or DF-DFID value.

Separate each line from the next with a comma, except for the last.

23.5.1.4 Generate the FROM table line.

If ORACLE, generate

FROM rtid

If DB2, generate

FROM user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

23.5.1.5 Perform steps 21.1.6 through 21.1.6.2.5.5 to generate the WHERE clause.

23.5.1.6 If ORACLE, generate the following line.

FOR UPDATE

23.5.1.7 If DB2, generate the following lines.

FOR UPDATE OF

Scan the IS-ACTION-LIST. For each entry which has both IS-DFNO and IS-RTNO not equal zero, IS-SUBTRANS-ID equal RPS-SUBTRANS, IS-DELETE-FLAG not equal 1, and IS-MAPPED-TO-FLAG = "Y", generate

user.rtid.dfid

where user is the DB-USER-ID value, rtid is the IS-RTID value and dfid is the IS-DFID value.

Separate each line from the next by a comma except for the last line.

23.5.1.8 Generate the END EXEC statement.

END-EXEC.

23.5.2 Generate the OPEN CURSOR statement.

EXEC SQL OPEN IISSCUR END-EXEC.

23.5.3 Generate the error checking logic by substituting, if ORACLE, KES-ORACLE-OPEN-ERROR for P1 and if DB2, KES-DB2-OPEN-ERROR for P1 and UNABLE TO OPEN CURSOR for P2 in macro CDQPS06.

23.5.4 Generate the following label.

FETCH-LOOP.

23.5.5 Initialize all receiving fields.

For each entry selected in step 23.5.1.3, generate the following 2 lines.

MOVE ZERO TO D-dfno-rtno.

MOVE ZERO TO INDP-dfno-rtno.

where dfno is either the IS-DFNO, ISQ-DFNOL, ISQ-DFNOR, CMA-DF-NO or DF-DFNO of the field selected and rtno is the corresponding IS-RTNO, ISQ-RTNOL, ISQ-RTNOR or CMA-RT-NO.

23.5.6 Generate the FETCH statement.

23.5.6.1 Generate the following line.

EXEC SQL FETCH IISSCUR INTO

23.5.6.2 Generate the receiving variables and indicators.

For each field selected in step 23.5.5, generate

:D-dfno-rtno:INDP-dfno-rtno

where dfno is either the IS-DFNO, ISQ-DFNOL, ISQ-DFNOR, CMA-DF-NO or DF-DFNO of the field selected and rtno is the corresponding IS-RTNO, ISQ-RTNOL, ISQ-RTNOR or CMA-RT-NO.

23.5.6.3 Generate the following line.

```
END-EXEC.
```

23.5.7 Generate FETCH error handling logic by substituting ORACLE or DB2 depending on the target database for P1 in macro CDQPS08.

23.5.8 Generate the internal to conceptual transform logic by closing the work file, calling CDIC with the following parameters and reopening the work file for EXTEND upon return.

```
INPUT
  WORK-FILE                          PIC X(30)
  RPS-SUBTRANS                        PIC 9(3)
  IS-ACTION-LIST
  COMPLEX-MAPPING-ALG-TABLE
  NUMERIC-NULL                       PIC X(30)
  CHAR-NULL                          PIC X(30)
  CS-QUALIFY-LIST
  IS-QUALIFY-LIST
OUTPUT
  RET-STATUS                          PIC X(5)
```

23.5.9 Generate the conceptual IF by closing the work file, calling CDRPCIF with the following parameters and reopening the file for EXTEND upon return.

```
INPUT
  BOOLEAN-LIST
  CS-QUALIFY-LIST
  CS-ACTION-LIST
  IS-QUALIFY-LIST
  WORK-FILE  PIC X(30)
OUTPUT
  NONE
```

23.5.10 Generate the IF termination by generating macro CDQPS13 which has no parameters.

23.5.11 Generate the UPDATE statement.

23.5.11.1 Generate the following line.

EXEC SQL

23.5.11.2 If ORACLE, generate the following line.

UPDATE rtid

If DB2, generate

UPDATE user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

23.5.11.3 Generate the SET keyword.

SET

23.5.11.4 Generate the column = variable portion of the UPDATE statement.

Scan the IS-ACTION-LIST. For each used entry with IS-SUBTRANS-ID equal RPS-SUBTRANS and IS-DELETE-FLAG not equal 1, test the IS-ALG-PTR. If the IS-ALG-PTR equal zero, generate

dfid

On the next line, generate

=

On the next line, generate

:IS-VAR-isindex:INDP-isindex

where dfid is the IS-DFID value and isindex is the IS-INDEX value.

If the IS-ALG-PTR does not equal zero, set CMA-INDEX to IS-ALG-PTR. Scan the CMA-PARM-ENTRY searching for CMA-DFID not equal spaces. Generate

dfid

On the next line, generate

=

On the next line, generate

:IS-mod-inst-pno:INDP-mod-inst-pno

where dfid is the IS-DFID value, mod is the IS-ALG-ID value, inst is the CMA-MOD-INST value pointed to by IS-ALG-PTR and pno is the IS-PARM-NO value with non-blank CMA-DFID.

Separate each 3 line entry from the next by a comma except for the last.

23.5.11.5 Generate the following two lines.

WHERE CURRENT OF IISSCUR
END-EXEC.

23.5.12 Generate the error checking logic by substituting, if ORACLE, KES-ORACLE-UPDATE-ERROR and if DB2, KES-DB2-UPDATE-ERROR for P1 and UNABLE TO UPDATE for P2 in macro CDQPS06.

23.5.13 Generate the following line.

GO TO FETCH-LOOP.

23.5.14 Generate the program end by generating macro CDQPS12 which has no parameters.

23.5.15 Go to step 25.

24. Generate the remainder of the subroutine for delete.

24.1 Scan the IS-ACTION-LIST searching for an entry with IS-MAPPED-TO-FLAG equal N and IS-DELETE-FLAG not equal 1. If found perform the following steps, otherwise, go to step 24.2.

24.1.1 For each IS-ACTION entry with IS-MAPPED-TO-FLAG equal Y, IS-SUBTRANS-ID

equal RPS-SUBTRANS, and IS-DELETE-FLAG not equal 1, set up either a native or user defined null value.

If the entry has IS-DATA-TYPE equal C and CHAR-NULL does not equal NULL, generate

```
MOVE charnull TO IS-VAR-isindex.
MOVE ZERO TO INDP-isindex.
```

where charnull is the CHAR-NULL value and isindex is the IS-INDEX value.

If the entry has IS-DATA-TYPE equal C and CHAR-NULL equal NULL or has IS-DATA-TYPE not equal C and NUMERIC-NULL equal NULL, generate

```
MOVE ZERO TO IS-VAR-isindex.
MOVE -1 TO INDP-isindex.
```

where isindex is the IS-INDEX value.

If the entry has IS-DATA-TYPE not equal C and NUMERIC-NULL not equal NULL, generate

```
MOVE numnull TO IS-VAR-isindex.
MOVE ZERO TO INDP-isindex.
```

where numnull is the NUMERIC-NULL value and isindex is the IS-INDEX value.

24.2 If all used ISQ-RTIDLs and non-blank ISQ-RTIDRs equal IS-RTID (1) (no USING clause) and all used ISQ-EVAL-FLAGs are not equal zero (everything is evaluatable internally), perform the following steps, otherwise go to step 24.3.

24.2.1 If any IS-MAPPED-TO-FLAG equal N, an update statement must be generated, otherwise go to step 24.2.2 to generate the delete statement.

24.2.1.1 Generate the EXEC SQL statement.

```
EXEC SQL
```

24.2.1.2 Generate the UPDATE statement.

If ORACLE, generate

UPDATE rtid

If DB2, generate

UPDATE user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

24.2.1.3 Generate the SET keyword.

SET

24.2.1.4 Generate the column = variable portion of the UPDATE statement.

Scan the IS-ACTION-LIST. For each used entry with IS-MAPPED-TO-FLAG equal Y, IS-SUBTRANS-ID equal RPS-SUBTRANS, and IS-DELETE-FLAG not equal 1, generate

dfid

=

:IS-VAR-isindex:INDP-isindex

where dfid is the IS-DFID value and isindex is the IS-INDEX value.

Separate each 3 line entry from the next by a comma except for the last entry.

24.2.1.5 Perform steps 21.1.6 through 21.1.6.2.5.5 to generate the WHERE clause.

24.2.1.6 Generate the END-EXEC statement.

END-EXEC.

24.2.1.7 Generate the error checking logic by substituting, if ORACLE, KES-ORACLE-DELETE-ERROR and if DB2, KES-DB2-DELETE-ERROR for P1 and UNABLE TO DELETE for P2 in macro CDQPS06A.

24.2.1.8 Generate the program end by generating macro CDQPS10 which has no parameters.

24.2.1.9 Go to step 25.

24.2.2 Generate the DELETE statement.

24.2.2.1 Generate the EXEC SQL statement.

EXEC SQL

24.2.2.2 Generate the DELETE

DELETE FROM

24.2.2.3 Generate the table name.

If ORACLE, generate

rtid

If DB2, generate

user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

24.2.3.4 Perform steps 21.1.6 through 21.1.6.2.5.5 to generate the WHERE clause.

24.2.3.5 Generate the END-EXEC statement.

END-EXEC.

24.2.3.6 Generate the error checking logic by substituting, if ORACLE, KES-ORACLE-DELETE-ERROR and if DB2, KES-DB2-DELETE-ERROR for P1 and UNABLE TO DELETE for P2 in macro CDQPS06A.

24.2.3.7 Generate the program end by generating macro CDQPS10 which has no parameters.

24.2.3.8 Go to step 25.

24.3 Generate the table lock statement.

EXEC SQL LOCK TABLE

If ORACLE, generate

rtid

If DB2, generate

user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

Generate

IN EXCLUSIVE MODE END-EXEC.

24.4 Generate status checking logic by substituting either ORACLE or DB2, depending on the target database for P1 and IS-RTID (1) for P2 in macro CDQPS11.

24.5 If at least 1 used ISQ-RTIDL or non-blank ISQ-RTIDR does not equal IS-RTID (1), a using clause has been employed. Perform the following step, otherwise go to step 24.6.

24.5.1 Generate the DECLARE CURSOR statement.

24.5.1.1 Generate the first line.

EXEC SQL DECLARE IISSCUR CURSOR FOR

24.5.1.2 Generate the SELECT keyword.

SELECT

24.5.1.3 Generate the column list.

Scan the IS-ACTION-LIST searching for all entries which have IS-SUBTRANS-ID equal RPS-SUBTRANS, IS-DELETE-FLAG not equal 1, and which have both IS-DFNO and IS-RTNO not equal zero and IS-MAPPED-TO-FLAG equal Y.

For each IS entry found, generate 1 line according to one of the following formats, separating each line from the next by a comma except for the last line.

If ORACLE, generate

```
rtid1.dfid1,
      .
      .
      .
rtidn.dfidn
```

If DB2, generate

```
user.rtid1.dfid1,
      .
      .
      .
user.rtidn.dfidn
```

where user is the DB-USER-ID value, rtid is the IS-RTID value and dfid is the IS-DFID value.

24.5.1.4 Generate the FROM keyword.

FROM

24.5.1.5 Generate the FROM table list.

For each unique IS-RTID, ISQ-RTIDL and ISQ-RTIDR which have their corresponding IS-SUBTRANS-ID, ISQ-SUBTRANS-IDL or ISQ-SUBTRANS-IDR equal RPS-SUBTRANS and, for the IS qualify entries, the ISQ-EVAL-FLAG not equal zero, and the IS action entries with their IS-DELETE-FLAG not equal 1, generate 1 line according to one of the following formats, separating each line from the next by a comma except for the last line.

For ORACLE, generate

rtid1,
.
.
.
rtidn

For DB2, generate

user.rtid1,
.
.
.
user.rtidn

where user is the DB-USER-ID value and rtid is the IS-RTID, ISQ-RTIDL or ISQ-RTIDR value.

24.5.1.6 Perform steps 21.1.6 through 21.1.6.2.5.5 to generate the WHERE clause.

24.5.1.7 If ORACLE, generate the following line.

FOR UPDATE

24.5.1.8 If DB2, generate the following lines.

FOR UPDATE OF

Scan the IS-ACTION-LIST. For each entry which has both IS-DFNO and IS-RTNO not equal zero, IS-SUBTRANS-ID equal RPS-SUBTRANS, IS-DELETE-FLAG not equal 1, and is MAPPED-TO-FLAG equal Y, generate

user.rtid.dfid

where user is the DB-USER-ID value, rtid is the IS-RTID value and dfid is the IS-DFID value.

Separate each line from the next by a comma except for the last line.

24.5.1.9 Generate the END EXEC statement.

END-EXEC.

24.5.2 Generate the OPEN CURSOR statement.

EXEC SQL OPEN IISSCUR END-EXEC.

24.5.3 Generate the error checking logic by substituting, if Oracle, KES-ORACLE-OPEN-ERROR for P1 and if DB2, KES-DB2-OPEN-ERROR for P1 and UNABLE TO OPEN CURSOR for P2 in macro CDQPS06.

24.5.4 Generate the following label.

FETCH-LOOP.

24.5.5 Initialize all receiving fields. Scan the IS-ACTION-LIST. For each unique, non-zero IS-DFNO/IS-RTNO combination with IS-MAPPED-TO-FLAG equal Y, IS-SUBTRANS-ID equal RPS-SUBTRANS, and IS-DELETE-FLAG not equal 1, generate the following two lines.

MOVE ZERO TO D-dfno-rtno.

MOVE ZERO TO INDP-dfno-rtno.

where dfno is the IS-DFNO value and rtno is the IS-RTNO value.

24.5.6 Generate the FETCH statement.

24.5.6.1 Generate the following line.

EXEC SQL FETCH IISSCUR INTO

24.5.6.2 For each entry selected in 24.5.5, generate the following line, separating each line from the next by a comma except for the last line.

:D-dfno-rtno:INDP-dfno-rtno

where dfno is the IS-DFNO value and rtno is the IS-RTNO value.

24.5.6.3 Generate the following line.

END-EXEC.

24.5.7 Generate FETCH error handling logic by substituting ORACLE or DB2 depending on the target database for P1 in macro CDQPS08.

24.5.8 If any IS-MAPPED-TO-FLAG equal N, an update statement must be generated, otherwise go to step 24.5.9 to generate the delete statement.

24.5.8.1 Generate the EXEC SQL statement.

EXEC SQL

24.5.8.2 Generate the UPDATE statement.

If ORACLE, generate

UPDATE rtid

If DB2, generate

UPDATE user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

24.5.8.3 Generate the SET keyword.

SET

24.5.8.4 Generate the column = variable portion of the UPDATE statement.

Scan the IS-ACTION-LIST. For each used entry with IS-MAPPED-TO-FLAG equal Y, IS-SUBTRANS-ID equal RPS-SUBTRANS, and IS-DELETE-FLAG not equal 1, generate

dfid

=

:IS-VAR-isindex:INDP-isindex

where dfid is the IS-DFID value and isindex is the IS-INDEX value.

Separate each 3 line entry from the next by a comma except for the last entry.

24.5.8.5 Generate the following line:

WHERE CURRENT OF IISSCUR

24.5.8.6 Generate the END-EXEC statement.

END-EXEC.

24.5.8.7 Generate the error checking logic by substituting, if ORACLE, KES-ORACLE-DELETE-ERROR and if DB2, KES-DB2-DELETE-ERROR for P1 and UNABLE TO DELETE for P2 in macro CDQPS06.

24.5.8.8 Generate the following line.

GO TO FETCH-LOOP.

24.5.8.9 Generate the program end by generating macro CDQPS12 which has no parameters.

24.5.8.10 Go to step 25.

24.5.9 Generate the DELETE statement.

24.5.9.1 Generate the EXEC SQL statement.

EXEC SQL

24.5.9.2 Generate the DELETE

DELETE FROM

24.5.9.3 Generate the table name.

If ORACLE, generate

rtid

If DB2, generate

user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

24.5.9.4 Generate the following line:

WHERE CURRENT OF IISSCUR

24.5.9.5 Generate the END-EXEC statement.

END-EXEC.

24.5.9.6 Generate the error checking logic by substituting, if ORACLE, KES-ORACLE-DELETE-ERROR and if DB2, KES-DB2-DELETE-ERROR for P1 and UNABLE TO DELETE for P2 in macro CDQPS06.

24.5.9.7 Generate the following line.

GO TO FETCH-LOOP.

24.5.9.8 Generate the program end by generating macro CDQPS12 which has no parameters.

24.5.9.9 Go to step 25.

24.6 Generate the remainder of the program for the delete with no using clause and complex mapping in the qualification.

24.6.1 Generate the DECLARE CURSOR statement.

24.6.1.1 Generate the first line.

EXEC SQL DECLARE IISSCUR CURSOR FOR

24.6.1.2 Generate the SELECT keyword.

SELECT

24.6.1.3 Generate the column list.

Scan the IS-ACTION-LIST searching for all entries which have IS-SUBTRANS-ID equal RPS-SUBTRANS and which have both IS-DFNO and IS-RTNO not equal zero and IS-MAPPED-TO-FLAG equal Y and IS-DELETE-FLAG not = 1.

In addition, scan the IS-QUALIFY list searching for non-zero ISQ-DFNOL/ISQ-RTNOL and ISQ-SUBTRANS-IDL

equal SUBTRANS-ID, and
ISQ-DFNOR/ISQ-RTNOR and ISQ-SUBTRANS-IDR
equal SUBTRANS-ID.

Scan the COMPLEX-MAPPING-ALG-TABLE
searching for all entries which have
CMA-SUTRANSACTION equal RPS-SUBTRANS.
For each subtrans satisfying the
argument, search for non-zero CMA-RT-NO.
If found, look for non-zero CMA-DF-NO.
If not found, scan CMA-OF-ENTRY searching
for DF-MOD-PTR equal CMA-INDEX and
DF-PARM-PTR equal CMA-PARM-INDEX.

For each IS-ACTION entry satisfying the
above arguments, as well as each unique
IS-QUALIFY entry or CMA entry, generate a
SELECT list. Generate no duplicates.

If ORACLE, generate

rtid.dfid

If DB2, generate

user.rtid.dfid

where user is the DB-USER-ID value, rtid
is either the IS-RTID, ISQ-RTIDL,
ISQ-RTIDR or CMA-RTID value and dfid is
either the IS-DFID, ISQ-DFIDL, ISQ-DFIDR,
CMA-DF-ID or DF-DFID value.

Separate each line from the next with a
comma, except for the last.

24.6.1.4 Generate the FROM table line.

If ORACLE, generate

FROM rtid

If DB2, generate

FROM user.rtid

where user is the DB-USER-ID value and
rtid is the IS-RTID (1) value.

24.6.1.5 Perform steps 21.1.6 through 21.1.6.2.5.5 to generate the WHERE clause.

24.6.1.6 If ORACLE, generate the following line.

FOR UPDATE

24.6.1.7 If DB2, generate the following lines.

FOR UPDATE OF

Scan the IS-ACTION-LIST. For each entry which has both IS-DFNO and IS-RTNO not equal zero, and IS-SUBTRANS-ID equal RPS-SUBTRANS, IS-DELETE-FLAG not equal 1, and IS-MAPPED-TO-FLAG equal Y, generate

user.rtid.dfid

where user is the DB-USER-ID value, rtid is the IS-RTID value and dfid is the IS-DFID value.

Separate each line from the next by a comma except for the last line.

24.6.1.8 Generate the END EXEC statement.

END-EXEC.

24.6.2 Generate the OPEN CURSOR statement.

EXEC SQL OPEN IISSCUR END-EXEC.

24.6.3 Generate the error checking logic by substituting, if ORACLE, KES-ORACLE-OPEN-ERROR for P1 and if DB2, KES-DB2-OPEN-ERROR for P1 and UNABLE TO OPEN CURSOR for P2 in macro CDQPS06.

24.6.4 Generate the following label.

FETCH-LOOP.

24.6.5 Initialize all receiving fields.

For each entry selected in step 24.6.1.3 generate the following 2 lines.

MOVE ZERO TO D-dfno-rtno.

MOVE ZERO TO INDP-dfno-rtno.

where dfno is either the IS-DFNO, ISQ-DFNOL, ISQ-DFNOR, CMA-DF-NO or DF-DFNO of the field selected and rtno is the corresponding IS-RTNO, ISQ-RTNOL, ISQ-RTNOR or CMA-RTNO.

24.6.6 Generate the FETCH statement.

24.6.6.1 Generate the following line.

EXEC SQL FETCH IISSCUR INTO

24.6.6.2 Generate the receiving variables and indicators.

For each field selected in step 24.6.5, generate

:D-dfno-rtno:INDP-dfno-rtno

where dfno is either the IS-DFNO, ISQ-DFNOL, ISQ-DFNOR, CMA-DF-NO or DF-DFNO of the field selected and rtno is the corresponding IS-RTNO, ISQ-RTNOL, ISQ-RTNOR or CMA-RT-NO.

24.6.6.3 Generate the following line.

END-EXEC.

24.6.7 Generate FETCH error handling logic by substituting ORACLE or DB2 depending on the target database for P1 in macro CDQPS08.

24.6.8 Generate the internal to conceptual transform logic by closing the work file, calling CDIC with the following parameters and reopening the work file for extend upon return.

```
INPUT
  WORK-FILE                          PIC X(30)
  RPS-SUBTRANS                        PIC 9(3)
  IS-ACTION-LIST
  COMPLEX-MAPPING-ALG-TALE
  NUMERIC-NULL                       PIC X(30)
  CHAR-NULL                          PIC X(30)
  CS-QUALIFY-LIST
  IS-QUALIFY-LIST
OUTPUT
  RET-STATUS                          PIC X(5)
```

24.6.9 Generate the conceptual IF by closing the work file, calling CDRPCIF with the following parameters and reopening the file for EXTEND upon return.

```
INPUT
  BOOLEAN-LIST
  CS-QUALIFY-LIST
  CS-ACTION-LIST
  IS-QUALIFY-LIST
  WORK-FILE                          PIC X(30)
OUTPUT
  NO PARAMETERS
```

24.6.10 Generate the IF termination by generating macro CDQPS13 which has no parameters.

24.6.11 If any IS-MAPPED-TO-FLAG equal N, an update statement must be generated, otherwise go to step 24.6.12 to generate the delete statement.

24.6.11.1 Generate the EXEC SQL statement.

EXEC SQL

24.6.11.2 Generate the UPDATE statement.

If ORACLE, generate

UPDATE rtid

If DB2, generate

UPDATE user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

24.6.11.3 Generate the SET keyword.

SET

24.6.11.4 Generate the column = variable portion of the UPDATE statement.

Scan the IS-ACTION-LIST. For each used entry with IS-MAPPED-TO-FLAG equal Y, IS-SUBTRANS-ID equal RPS-SUBTRANS, and IS-DELETE-FLAG not equal 1, generate

dfid

=

:IS-VAR-isindex:INDP-isindex

where dfid is the IS-DFID value and isindex is the IS-INDEX value.

Separate each 3 line entry from the next by a comma except for the last entry.

24.6.11.5 Generate the following line.

WHERE CURRENT OF IISSCUR

24.6.11.6 Generate the END-EXEC statement.

END-EXEC.

24.6.11.7 Generate the error checking logic by substituting, if ORACLE, KES-ORACLE-DELETE-ERROR and if DB2, KES-DB2-DELETE-ERROR for P1 and UNABLE TO DELETE for P2 in macro CDQPS06.

24.6.11.8 Generate the following line.

GO TO FETCH-LOOP.

24.6.11.9 Generate the program end by generating macro CDQPS12 which has no parameters.

24.6.11.10 Go to step 25.

24.6.12 Generate the DELETE statement.

24.6.12.1 Generate the EXEC SQL statement.

EXEC SQL

24.6.12.2 Generate the DELETE

DELETE FROM

24.6.12.3 Generate the table name.

If ORACLE, generate

rtid

If DB2, generate

user.rtid

where user is the DB-USER-ID value and rtid is the IS-RTID (1) value.

24.6.12.4 Generate the following line.

WHERE CURRENT OF IISSCUR

24.6.12.5 Generate the END-EXEC statement.

END-EXEC.

24.6.12.6 Generate the error checking logic by substituting, if ORACLE, KES-ORACLE-DELETE-ERROR and if DB2, KES-DB2-DELETE-ERROR for P1 and UNABLE TO DELETE for P2 in macro CDQPS06.

24.6.12.7 Generate the following line.

GO TO FETCH-LOOP.

24.6.12.8 Generate the program end by generating macro CDQPS12 which has no parameters.

24.6.12.9 Go to step 25.

25. Close the work file and terminate processing.

23.5 Outputs

GEN-FILE-NAME PIC X(30)

GEN-FILE-NAME will contain the name of the file which will contain the generated request processor.

RET-STATUS PIC X(5)

RET-STATUS will contain CDQPS's completion status. A value equal to KES-SUCCESSFUL as defined in the ERRCDM copy member indicates success.

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

LIBRARY NAME - SQL
MACRO NAME - CDQPS01

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  P1.
ENVIRONMENT DIVISION.
DATA DIVISION.
```

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

LIBRARY NAME - SQL
MACRO NAME - CDQPS02

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  P1.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
COPY ERRCDM OF IISSCLIB.
01  RET-STATUS       PIC X(5).
01  MODULE-NAME      PIC X(10) VALUE "P1".
01  MESG-DESC        PIC X(60).
01  SHO-CODE         PIC -----9.
01  NO-MORE-DATA     PIC S9(4) COMP VALUE P2.
01  LOCAL-NULL-FLAG  PIC 9.
01  RECORD-LENGTH    PIC S9(9) COMP.
```

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

LIBRARY NAME - SQL
MACRO NAME - CDQPS03

```
COPY ERRCDM OF IISSCLIB.
COPY ERRFS OF IISSCLIB
01  RET-STATUS           PIC X(5).
01  MODULE-NAME          PIC X(10) VALUE "P1".
01  MESG-DESC            PIC X(60).
01  SHO-CODE             PIC -----9.
01  RESFILE              PIC X(30).
01  MY-HOST              PIC XXX VALUE SPACES.
01  FILE-OPEN            PIC 9.
    88 FILE-IS-OPEN VALUE 1.
01  LOCAL-NULL-FLAG      PIC 9.
01  NO-MORE-DATA         PIC S9(4) COMP VALUE P2.
01  FCB1                 PIC S9(9) COMP.
01  RECORD-LENGTH        PIC S9(9) COMP.
01  ACCESS-MODE          PIC X.
01  NUMBER-OF-RECORDS    PIC S9(9) COMP VALUE 2000.
01  ERROR-FLAG           PIC X(5).
```

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

LIBRARY NAME - SQL
MACRO NAME - CDQPS04

```
01  MESSAGE-BODY-OUT.
     03 OUTFILE-NAME     PIC X(30).
     03 REC-COUNT        PIC 9(6).
    03 RP-STATUS        PIC X(5).
PROCEDURE DIVISION USING MESSAGE-BODY-IN
                         MESSAGE-BODY-OUT.
START HERE.
    MOVE KES-SUCCESSFUL TO RP-STATUS.
    MOVE KES-SUCCESSFUL TO RET-STATUS.
    MOVE ZERO TO REC-COUNT.
    MOVE SPACES TO OUTFILE-NAME.
    MOVE SPACES TO MESG-DESC.
    MOVE ZERO TO LOCAL-NULL-FLAG.
```

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

LIBRARY NAME - SQL
MACRO NAME - CDQPS05

```
MOVE ZERO TO FILE-OPEN.
CALL "NAMFIL" USING RESFILE.
 IF RESFILE = LOW-VALUES
    MOVE "UNABLE TO GENERATE RESULTS FILE"
          TO MESG-DESC
    MOVE KES-NOFILENAME TO RET-STATUS
    PERFORM PROCESS-ERROR
    MOVE RET-STATUS TO RP-STATUS
    GO TO EXIT-PROGRAM.
MOVE RESFILE TO OUTFILE-NAME.
```

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

LIBRARY NAME - SQL
MACRO NAME - CDQPS06A

```
IF SQLCODE NOT = ZERO AND SQLCODE NOT = NO-MORE-DATA
    MOVE SQLCODE TO SHO-CODE
    MOVE P1 TO RET-STATUS RP-STATUS
    STRING "P2, CODE IS"
          SHO-CODE DELIMITED BY SIZE INTO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
```

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

```
LIBRARY NAME - SQL
  MACRO NAME - CDQPS07


   MOVE "W" TO ACCESS-MODE.
   CALL "OPNFIL" USING
                 FCB1,
                 ERROR-FLAG,
                 RESFILE,
                 ACCESS-MODE,
                 RECORD-LENGTH,
                 NUMBER-OF-RECORDS.
   IF ERROR-FLAG NOT = KES-FILE-OK
      STRING "RESFILE OPEN ERROR: " ERROR-FLAG
             DELIMITED BY SIZE INTO MESG-DESC
      MOVE KES-OPEN-NOT-SUCCESSFUL TO RET-STATUS
      GO TO EXIT-PROGRAM.
   MOVE 1 TO FILE-OPEN.
FETCH-LOOP.
```

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

```
LIBRARY NAME - SQL
  MACRO NAME - CDQPS08


   IF SQLCODE NOT = ZERO
        IF SQLCODE = NO-MORE-DATA
             GO TO EXIT-PROGRAM
        ELSE
             MOVE SQLCODE TO SHO-CODE
             MOVE KES-P1-FETCH-ERROR TO RET-STATUS RP-STATUS
             STRING "UNABLE TO FETCH, CODE IS"
                 SHO-CODE DELIMITED BY SIZE INTO MESG-DESC
             PERFORM PROCESS-ERROR
              GO TO EXIT-PROGRAM.
```

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

```
LIBRARY NAME - SQL
  MACRO NAME - CDQPS09


   CALL "OUTFIL" USING
                 FCB1,
                 ACCESS-MODE,
                 RESULTS-REC,
                 RECORD-LENGTH.
   IF ERROR-FLAG NOT = KES-FILE-OK
      STRING "RESULTS-REC WRITE ERROR: " ERROR-FLAG
             DELIMITED BY SIZE INTO MESG-DESC
      MOVE KES-WRITE-NOT-SUCCESSFUL TO RET-STATUS
      GO TO EXIT-PROGRAM.
    ADD 1 TO REC-COUNT.
   GO TO FETCH-LOOP.
COPY ERRPRO OF IISSCLIB.
EXIT-PROGRAM.
    IF FILE-IS-OPEN
      MOVE "K" TO ACCESS-MODE
       CALL "CLSFIL" USING
                     FCB1,
                     ERROR-FLAG,
                     ACCES-MODE
   IF ERROR-FLAG NOT = KES-FILE-OK
      STRING "RESFILE CLOSE ERROR: " ERROR-FLAG
             DELIMITED BY SIZE INTO MESG-DESC
      MOVE KES-CLOSE-NOT-SUCCESSFUL TO RET-STATUS.
   EXEC SQL CLOSE IISSCUR END-EXEC.
   IF SQLCODE NOT = ZERO
      MOVE SQLCODE TO SHO-CODE
      MOVE KES-CLOSE-CURSOR TO RET-STATUS RP-STATUS
      STRING "UNABLE-TO-CLOSE-CURSOR, CODE IS"
      SHO-CODE DELIMITED BY SIZE INTO MESG-DESC
      PERFORM PROCESS-ERROR.
EXIT2.
     EXIT PROGRAM.
```

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

LIBRARY NAME - SQL
MACRO NAME - CDQPS10

```
EXIT-PROGRAM.
     EXIT PROGRAM.
COPY ERRPRO OF IISSCLIB.
```

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

LIBRARY NAME - SQL
MACRO NAME - CDQPS11

```
   IF SQLCODE NOT = ZERO
        MOVE SQLCODE TO SHO-CODE
        MOVE KES-P1-LOCK-ERROR TO RET-STATUS RP-STATUS
        STRING "UNABLE TO LOCK " DELIMITED BY SIZE
             "P2" DELIMITED BY SPACE
             " CODE" DELIMITED BY SIZE
             SHO-CODE DELIMITED BY SIZE INTO MESG-DESC
        PERFORM PROCESS-ERROR
        GO TO EXIT-PROGRAM.
```

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

```
LIBRARY NAME - SQL
  MACRO NAME - CDQPS12


EXIT-PROGRAM.
   EXEC SQL CLOSE IISSCUR END-EXEC.
   IF SQLCODE NOT = ZERO
      MOVE SQLCODE TO SHO-CODE
      MOVE KES-CLOSE-CURSOR TO RET-STATUS RP-STATUS
      STRING "UNABLE-TO-CLOSE-CURSOR, CODE IS"
         SHO-CODE DELIMITED BY SIZE INTO MESG-DESC
      PERFORM PROCESS-ERROR.
EXIT-2.
      EXIT-PROGRAM.
COPY ERRPRO OF IISSCLIB.
```

TABLE 23-1

SQL REQUEST PROCESSOR MACROS

LIBRARY NAME - SQL
MACRO NAME - CDQPS13

```
            NEXT SENTENCE
        ELSE
            GO TO FETCH-LOOP.
```


☆U.S. Government Printing Office:1992—648-127/62427