INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 9 - Neutral Data Manipulation Language (NDML) Precompiler
Development Specification
Section 2 of 5

J. Althoff, M. Apicella

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

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

Approved for Public Release; Distribution is Unlimited

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

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

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

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

FOR THE COMMANDER:

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

If your address has changed, if you wish to be removed from 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**

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. **AGENCY USE ONLY** (Leave Blank)

2. **REPORT DATE**
   September 1990

3. **REPORT TYPE AND DATES COVERED**
   Final Technical Report
   1 Apr 87 - 31 Dec 90

4. **TITLE AND SUBTITLE**
   INTEGRATED INFORMATION SUPPORT SYSTEM (IIS)
   Volume V: Common Data Model Subsystem
   Part 9: Neutral Data Manipulation Language (NDML) Precompiler Development Specification
   Section 2 of 5

5. **FUNDING NUMBERS**
   Contract No.: F33600-87-C-0464
   PE: 78011F
   Proj. No.: 595600
   Task No.: F95600
   WU: 20950607

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

7. **PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
   Control 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**
    Section 2 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**
    SAR

18. **SECURITY CLASSIFICATION OF THIS PAGE**
    SAR

19. **SECURITY CLASSIFICATION OF ABSTRACT**
    SAR

20. **LIMITATION ABSTRACT**
    SAR

Standard Form 298 (Rev 2-88)
Prescribed by ANSI Std Z239-18
298-102
SECTION 16

FUNCTION PRE6 - SELECT IS ACCESS PATH

The IS Access Path Selector is a compile-time module whose purpose is to transform a Subtransaction identified by PRE5 - Decompose CS NDML into an access path for traversal through the appropriate local database. Each Subtransaction accesses only one database, managed by one DBMS, at one computer. There may be several Subtransactions that access the same database. Results of Subtransactions are joined or unioned by the Aggregator CI. PRE6 is called by PRE5.

The IS Access Path Selector derives access paths for databases managed by CODASYL and TOTAL databases. It is bypassed for Subtransactions that access relational databases.

Access paths for relational databases are provided by their DBMSs. For relational databases, the NDML of a Subtransaction is transformed to the DML of the relational DBMS by the Request Process Generator that handles the Subtransaction. In effect, the NDML serves as the generic relational DML, removing the need to use PRE7 - Transform IS Access Path to Generic DML, as well as PRE6 - Select IS Access Path.

The IS Access Path Selector finds the "best" access path through the internal schema, where "best" is considered to be the path that has either a calc key port or the fewest "find member of set" and "find next of area" commands.

The IS Access Path Selector will find only paths that conform to certain rules, making them confluent hierarchies. A confluent hierarchy is built of hierarchies joined by a common base record type. A common hierarchy of two record sets (e.g. A owns B and B owns C) is a degenerate confluent hierarchy. The most basic non-degenerate confluent hierarchy is formed when a record type is a member in more than one record set (e.g., A owns B and C owns B). By contrast, the most basic form of access path structure that violates the rules for a confluent hierarchy is formed when a record type is an owner in more than one record set (e.g., A owns B and A owns C).

A confluent hierarchy can have any number of levels, but no record type can be an owner in more than one record set. Any record type may be a member in multiple record sets.

Conformance to confluent hierarchy rules is required only for Subtransaction access paths. An internal schema certainly does not have to be a confluent hierarchy, nor does a
Transaction's access path. The Aggregator CI will join/union results of the Subtransactions to form Transaction results. Note also that the confluent hierarchy rules for Subtransaction access paths are the same as the rules for forming proper external schemas by projects and joins from the conceptual schema.

Any record type in an access path is a candidate for the entry point for database access, i.e., for the "port" of the access. From a candidate port, the IS Access Path Selector searches the surrounding set structure to find all the referenced record types. To ensure adherence to the rules of confluent hierarchies, once a path starts "up", it cannot proceed down from a record type other than the port.

The IS Access Path Selector performs the following sequence of steps:

* It receives a relational Subtransaction from the Decomposer (PRE5).

* For each candidate key port (identified for insert by record key = variable, and for select, modify, or delete by a where clause in the form of field = variable, where the field is a record key), it does the following:
  * Selects a unique key port in preference to a duplicate key port.
  * Selects the key port with the greatest number of "find owner of set" commands.
  * It creates an access path based on information in the IS-ACTION-LIST, IS-QUALIFY-LIST, and SET-TABLE.

* If there are no candidate key ports, it performs activities similar to those of a key port to create an access path that is built upwards, starting with the record type at the bottom of the set chain for the subtransaction.

* A non-key port is chosen only if there are no possible key ports.

* It packages the access path for use by PRE7.

16-2
16.1 Inputs

1. CDM Metadata

The entity classes needed are:

- Component Data Field = CDF (E195)
- Database = DB (E24)
- Database Area Assignment = DBAA (E103)
- Data Field = DF (E67)
- DBMS on Host = DBMS on Host (E20)
- Record Set = RS (E72)
- Record Type = RT (E66)

2. The NDML internal schema request for which the access path is to be selected. The request is in the form of:

   IS-QUALIFY-LIST
   SET-TABLE
   OCCURS-TABLE-FOR-PRE7
   COMPLEX-MAPPING-ALGORITHM-TABLE

which is output from function PRE5.

3. The parenthesized logic to be applied to each subtransaction, along with the conditions which can be evaluated at the internal schema level. The information is contained in:

   SUBTRANS-BOOLEAN-LIST

which is output from function PRE5A.

16.2 Processing

1. Receive the Subtransaction (IS-ACTION-LIST, IS-QUALIFY-LIST, OCCURS-TABLE, and SET-TABLE) from PRE5-Decomposer. All NDML-NOs, DBNOs in the Subtransaction must be identical, thus we generally omit any further reference to DBNO, NDML-NO when naming records or fields. The IS-ACTION-LIST entries for a Subtransaction will all have the same IS-ACTION value.

   1a. Determine whether the nested repeating data fields, if any, conflict with the record sets that are involved in the subtransaction.

Find the OT7-SUBTRANS-ID entry for this subtransaction.
in the OCCURS-TABLE-FOR-PRE7. If one is not found, go to Step 2.

Find all the SET-TABLE entries for this subtransaction. If none are found, go to Step 2.

If OT7-RTNO in the OT7-SUBTRANS entry = ST-OWNER in any of the SET-TABLE entries, reject the NDML statement (repeating data fields in set owners cannot be accessed).

2. Identify candidate key ports for all types of actions that traverse the database (including selects, inserts, modifies, and deletes) by doing the following:

2.1 If IS-ACTION = 'I'
then for each non-blank IS-RTNO in the IS-ACTION-LIST:
group the IS-ACTION-LIST entries with that IS-RTNO
else for each non-blank ISQ-RTNOL in the IS-QUALIFY-LIST:
group the IS-QUALIFY-LIST entries with that ISQ-RTNOL and ISQ-TYPE = '2' and ISQ-OP = '='.

2.2 Determine if all record key members are represented in the qualify list:

Note - If IS-ACTION = 'I', make the following substitutions in Steps 2.2.1 through 2.2.4:

<table>
<thead>
<tr>
<th>IS-DBNO</th>
<th>ISQ-DBNOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS-RTNO</td>
<td>ISQ-RTNOL</td>
</tr>
<tr>
<td>IS-DFNO</td>
<td>ISQ-DFNOL</td>
</tr>
<tr>
<td>IS-INDEX</td>
<td>ISQ-INDEX</td>
</tr>
</tbody>
</table>

2.2.1 For each group identified in Step 2.1:

Find all the DF (E67) entries with DBNO, RTNO = ISQ-DBNOL, ISQ-RTNOL for the group and with RECORD-KEY-CODE = 'U' or 'D'.

2.2.2 For each DF entry found in Step 2.2.1, starting with those whose RECORD-KEY-CODE = 'U':

Note - In this step if IS-ACTION = 'I', 16-4
consider only the IS-ACTION-ENTRYs that have IS-MAPPED-TO-FLAG = 'Y', i.e. only those for which values will be provided.

Determine if it is in the group by checking for DFNO = ISQ-DFNOL among only those list entries in the group. If it is in the group, go to Step 2.2.3.

If it is not in the group, determine whether all its components, if any, are in the group by searching the hierarchy of component data fields. This search proceeds from the DF entry to any CDF (E195) entries with the same DBNO, RTNO, Group DFNO as the DF entry, and then to the DF entries that have DBNO, RTNO, DFNO = DBNO, RTNO, Comp DFNO of the CDF entries. The search continues iteratively until no more CDF entries are found.

If a DF entry (at any level) is found with DFNO = ISQ-DFNOL among the list entries in the group perform Step 2.2.3, and continue the search with the next branch of the hierarchy, i.e., do NOT check components of a DF entry that is found in the group.

If a DF entry is not found among the list entries in the group and if it does not have and CDF entries of its own, the original key data field is not completely represented in the list and so, cannot be used as a candidate key port. Abandon the search and remove any entries that were placed in the RECORD-KEY-TABLE. Repeat Step 2.2.2 for the next DF entry from Step 2.2.1.

If the search finishes without being abandoned, the original key data field is completely represented in the list and can be used as a candidate key port. Go to Step 2.2.4.

2.2.3 Build an RT-DATA-FIELDS entry in the RECORD-KEY-TABLE as follows:

RK-DFID = DFNAME in the DF entry
that matches the list entry found in Step 2.2.2

RK-DFNO
DFNO in DF entry that matches the list entry found in Step 2.2.2

RK-ISQ-PTR = ISQ-INDEX of the list entry found in Step 2.2.2

2.2.4 Finish an RK-REC-KEY entry in the RECORD-KEY-TABLE as follows:

RK-RTID = ISQ-RTIDL for this group
RK-RTNO = ISQ-RTNOL for this group
RK-DF-USED = Number of data fields in this key
RK-KEY-CODE = RECORD-KEY-CODE in the DF entry

Note: U = unique key
D = duplicate key

Repeat Step 2.2 for the next group.

2.3 If no candidate key ports were identified in Step 2.2, go to Step 2b.

2.4 Determine which key port will be the start of the access path by doing the following, first for the U's, then for the D's. A unique key port always takes precedence over a duplicate key port.

2.4.1 If there are no entries in the SET-TABLE, then there is only one record involved in this subtransaction. Select the first entry in the RECORD-KEY-TABLE as the key.

2.4.2 For each key represented in the RECORD-KEY-TABLE:

Key with ISQ-EVAL-FLAG > 2 cannot be used as key port

Search the SET-TABLE for an entry where ST-OWNER = RK-RTID
Using ST-OWNER as the starting point, traverse the set chain upwards, tallying the number of sets in the owner/member chain. Keep in RK table.

2.4.3 Select the key with the highest tally as the starting point in the access path.

2b. Identify the type of access path to be built by assigning a CASE-TYPE to the subtransaction as follows:

2b.1 Set CASE-TYPE = 1 if the following conditions are true:

1. No IS-QUALIFY-LIST entries whose ISQ-TYPE = 2 and whose ISQ-TYPE2-SOURCE = 'E' or 'I' are represented in the SUBTRANS-BOOLEAN-LIST.

2b.2 Set CASE-TYPE = 2 if the following conditions are true:

1. The RECORD-KEY-TABLE is empty
2. All IS-QUALIFY-LIST entries whose ISQ-TYPE = 2 and whose ISQ-TYPE2-SOURCE = 'E' or 'I' are ANDed. There must be at least 1 ISQ-EVAL-FLAG = 1 and no ISQ-EVAL-FLAG values > 1.

2b.3 Set CASE-TYPE = 3 if the following conditions are true:

1. A key port was selected in Step 2.4. We have a KEY-PORT-NO-ID.
2. Only one value for the key is represented in the IS-QUALIFY-LIST.
3. No IS-QUALIFY-LIST entries whose ISO-TYPE = 2 and whose ISQ-TYPE2-SOURCE = 'E' or 'I' are ORed between record types. There must be at least 1 ISQ-EVAL-FLAG = 1 and none > 1.

2b.4 Set CASE-TYPE = 4 if the following conditions are true:

1. A key port was selected in Step 2.4
2. Multiple values for the key are represented in the IS-QUALIFY-LIST. Search IS-QUALIFY for entry where type = 2E or I and ISQ-RTNOL = KEY-PORT-NO and ISQ-EVAL-FLAG > 1.
3. No IS-QUALIFY-LIST entries whose ISQ-TYPE = 2 and whose ISQ-TYPE2-SOURCE = 'E' or 'I' are ORed
between record types. There must be no
ISQ-EVAL-FLAG values greater than 3.

2b.5 Set CASE-TYPE = 5 if the following conditions are true:

1. The RECORD-KEY-TABLE is empty
2. No IS-QUALIFY-LIST entries whose ISQ-TYPE = 2 and
   whose ISQ-TYPE2-SOURCE = 'E' or 'I' are ORed
   between record types. There must be no
   ISQ-EVAL-FLAG > 3.

2b.6 Set CASE-TYPE = 6 if the following condition is true:

1. There exists in the ISQ-QUALIFY-LIST entries
   whose ISQ-TYPE = 2 and whose ISQ-TYPE2-SOURCE =
   'E' or 'I', which are ORed between record types.

2c. Generate access specifications to transform search
values to internal schema format.

For each IS-QUALIFY-LIST entry with

   ISQ-TYPE = '2' and
   ISQ-TYPE2-SOURCE = 'E' or 'I' and
   ISQ-ALG-IDL = blank

Write an MVS access specification:

ACCESS-TYPE = 'MVS'
MVS-ISQ-PTR = ISQ-INDEX

3. If CASE-TYPE = 3 or 4, use the key port identified in Step
   2.4 as the start of the access path by doing the following:

3.1 Set CURR-REC = RK-RTID
   CURR-RTNO = RK-RTNO

3.2 If CASE-TYPE = 3
   Write an 'RK' access specification:
   ACCESS-TYPE = RK'
   REC-SELECT-SPEC-PTR = RK-INDEX

   Set ISQ-LEFT = 1 for the IS-QUALIFY-LIST entry
   pointed to by RK-ISQ-PTR

3.3 If CASE-TYPE = 4

16-8
3.3.1 Write an 'RK1' access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} & = \text{'RK1'} \\
\text{RK1-LOOP-MAX} & = \text{number of entries in the IS-QUALIFY-LIST where} \\
& \text{ISQ-RTIDL} = \text{RK-RTID and} \\
& \text{ISQ-DFIDL} = \text{RK-DFID and} \\
& \text{ISQ-TYPE} = 2 \text{ and} \\
& \text{ISQ-TYPE2-SOURCE} = \text{'}E' \text{ and} \\
& \text{ISQ-SUBTRANS-IDL} \leq \text{SUB-ID}
\end{align*}
\]

3.3.2 For each entry in the IS-QUALIFY-LIST where:

\[
\begin{align*}
\text{ISQ-SUBTRANS-IDL} & = \text{SUB-ID} \\
\text{ISQ-RTIDL} & = \text{RK-RTID} \\
\text{ISQ-DFIDL} & = \text{RK-DFID} \\
\text{ISQ-TYPE} & = 2 \text{ AND} \\
\text{ISQ-TYPE2-SOURCE} & = \text{'}E'
\end{align*}
\]

Write an RK2 access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} & = \text{'RK2'} \\
\text{RK2-RK-INDEX} & = \text{RK-INDEX} \\
\text{RK2-LOOP-COUNT} & = \text{incremental count} \\
\text{RK2-DFID} & = \text{RK-DFID}
\end{align*}
\]

3.3.3 Write an RK3 access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} & = \text{'RK3'} \\
\text{REC-SELECT-SPEC-PTR} & = \text{RK-INDEX}
\end{align*}
\]

4. If CASE-TYPE = 1, 2, 5, or 6 generate an area search access path.

If the DBMS does not support area searches then issue an error message and stop. If the DBMS does support area searches then issue a warning message and continue.

4.1 Select the RTNO in the IS-ACTION-LIST or IS-QUALIFY-LIST that appears in the SET-TABLE at least once as a ST-MEMBER, but never as a ST-OWNER. If the SET-TABLE is empty, then only one RTNO appears in the IS-ACTION-LIST and IS-QUALIFY-LIST; that is the one to use.

Set CURR-REC = the port RTID  
CURR-RTNO = the port RTNO

16-9
4.2 This step was removed.

4.3 This step was removed.

4.4 Determine in which database areas the candidate non-key port resides:

Find the DBAA (E103) entries with RTNO = the candidate RTNO. Record the AREA IDs of the located entries.

4.5 Select one of the AREA IDs recorded in Step 4.4:

4.5.1 If IS-ACTION = 'S', '1', '2', 'K', 'M' or 'D'

Write an RA access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} & = 'RA' \\
\text{RAS-RTID} & = \text{CURR-REC} \\
\text{RAS-AREAID} & = \text{AREA ID}
\end{align*}
\]

4.5.2 If IS-ACTION = 'I'

Write an RAI access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} & = 'RAI' \\
\text{RAS-RTID} & = \text{CURR-REC} \\
\text{RAS-AREAID} & = \text{AREA ID}
\end{align*}
\]

4.6 Clear the GROUP-TABLE

4a. Determine if any conditions in the IS-QUALIFY-LIST for this subtransaction participate in complex mapping algorithms.

Search the IS-QUALIFY-LIST for an entry where

\[
\begin{align*}
\text{ISQ-TYPE} & = 2 \text{ or } 3 \text{ and} \\
\text{ISQ-EVAL-FLAG} & = 0 \text{ and} \\
\text{ISQ-SUBTRANS-IDL} & = \text{SUBTRANS-ID or} \\
\text{ISQ-SUBTRANS-IDR} & = \text{SUBTRANS-ID and} \\
\text{ISQ-ALG-IDL} & = \text{blank or} \\
\text{ISQ-ALG-IDR} & = \text{blank}
\end{align*}
\]

If an entry is found:

Set CMA-FLAG = 'Y'

5. Generate access specifications to process the current record by doing the following:

16-10
5.0a Generate access specification to move the current record from the schema area to working-storage.

Write an MR1 access specification:

```
ACCESS-TYPE = 'MR'
MR-RTNO = CURR-RTNO
MR-RTID = CURR-REC
```

5.a Generate access specifications to convert retrieved IS data values to CS format using complex mapping algorithms.

For each COMPLEX-MAPPING-ALGORITHM-TABLE entry with

```
CMA-SUBTRANSACTION = SUB-ID
CMA-RETR-UPD = 'R'
```

5.a.1 Generate access specifications to move entire records to algorithm input parameters.

For each CMA-PARAMETER-ENTRY with

```
CMA-RT-NO = CURR-RTNO and
CMA-DF-NO not filled in:
```

write an FU4 access specification:

```
ACCESS-TYPE = 'FU4'
FU4-ALG-ID = CMA-MOD-ID
FU4-MOD-INST = CMA-MOD-INSTANCE
FU4-PARM-NO = CMA-PARM-NO
FU4-RTID = CURR-REC
```

5.a.2 Generate access specifications to move data fields to algorithm input parameters.

For each CMA-PARAMETER-ENTRY with

```
CMA-RT-NO = CURR-RTNO and
CMA-DF-NO filled in:
```

write an FU3 access specification:

```
ACCESS-TYPE = 'FU3'
FU3-DFNO = CMA-DF-NO
FU3-ALG-ID = CMA-MOD-ID
FU3-PARM-NO = CMA-PARM-NO
```

16-11
5.a.3 Generate access specifications to move constant values to algorithm parameters.

For each CMA-PARAMETER-ENTRY with CMA-CONST-VAL filled in:

write an FG4 access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} & = \text{'FG4'} \\
\text{FG4-CONSTANT} & = \text{CMA-CONST-VAL} \\
\text{FG4-ALG-ID} & = \text{CMA-MOD-ID} \\
\text{FG4-MOD-INST} & = \text{CMA-MOD-INST} \\
\text{FG4-PARM-NO} & = \text{CMA-PARM-NO}
\end{align*}
\]

5.a.4 Generate access specifications to call complex mapping algorithms.

write a CAL access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} & = \text{'CAL'} \\
\text{CAL-ALG-ID} & = \text{CMA-MOD-ID} \\
\text{CAL-PARM-COUNT} & = \text{CMA-PARM-COUNT} \\
\text{CAL-MOD-INST} & = \text{CMA-MOD-INST}
\end{align*}
\]

5.a.5 Generate access specifications to move output algorithm parameters to CS tags.

If IS-ACTION = 'D' or 'M':

For each CMA-PARAMETER with CMA-TAG-NO filled in:

write an OU4 access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} & = \text{'OU4'} \\
\text{OU4-ALG-ID} & = \text{CMA-MOD-ID} \\
\text{OU4-MOD-INST} & = \text{CMA-MOD-INST} \\
\text{OU4-PARM-NO} & = \text{CMA-PARM-NO} \\
\text{OU4-TAG-NO} & = \text{CMA-TAG-NO}
\end{align*}
\]

5.b Generate access specifications to check record union discriminator predicates of where clause entries for all CASE-TYPE values, except CASE-TYPE = 6.

16-12
Search the IS-QUALIFY-LIST for an entry where

\[
\begin{align*}
\text{ISQ-RTIDL} &= \text{CURR-REC} \text{ and } \\
\text{ISQ-TYPE} &= 2 \text{ and } \\
\text{ISQ-TYPE2-SOURCE} &= 'U'
\end{align*}
\]

Write a UIF access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} &= 'UIF' \\
\text{UIF-RTNO} &= \text{CURR-RTNO}
\end{align*}
\]

NOTE: The UIF access type generates a call to a support routine which formats record union discrimination checks based on information in the SUBTRANS-BOOLEAN-LIST.

5.c Generate access specifications to check field-op-variable predicates of where clause entries where CASE-TYPE = 3, 4 or 5

5.c.1 For each IS-QUALIFY-LIST entry where

\[
\begin{align*}
\text{ISQ-RTNOL} &= \text{CURR-RTNO} \text{ and } \\
\text{ISQ-TYPE} &= 2 \text{ and } \\
\text{ISQ-TYPE2-SOURCE} &= 'E' \text{ or } 'I' \text{ and } \\
\text{ISQ-LEFT} &= 'N' \text{ and } \\
\text{ISQ-ALG-IDL} &= \text{blank and } \\
\text{ISQ-EVAL-FLAG} &= > 0
\end{align*}
\]

5.c.1.1 Set ISQ-LEFT = 1

5.c.1.2 If ISQ-EVAL-FLAG = 2 or 3

Write a RS5 access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} &= 'RS5' \\
\text{RS5-DFNO} &= \text{ISQ-DFNOL} \\
\text{RS5-OP} &= \text{ISQ-OP} \\
\text{RS5-ISQ-PTR} &= \text{ISQ-INDEX} \\
\text{RS5-SIDE} &= 'L' \\
\text{RS5-DF-TYPE} &= \text{ISQ-TYPEL} \\
\text{RS5-IF-OR} &= 'IF' \text{ for first RS5 access specification written for CURR-REC } \\
&\text{ 'OR' for second thru nth access specification written for CURR-REC}
\end{align*}
\]
5.c.1.3 If ISQ-EVAL-FLAG = 1

Write a RS4 access specification:

\[
\text{ACCESS-TYPE} = 'RS4', \\
\text{RS4-DFNO} = \text{ISQ-DFNOL}, \\
\text{RS4-OP} = \text{ISQ-OP}, \\
\text{RS4-ISQ-PTR} = \text{ISQ-INDEX}, \\
\text{RS4-SIDE} = 'L', \\
\text{RS4-DF-TYPE} = \text{ISQ-TYPEL}.
\]

5.c.2 If a RS5 access specification was written in Step 5.c.1.2

Write a NXS access specification:

\[
\text{ACCESS-TYPE} = 'NXS'.
\]

5.1 Generate access specifications to check field-op-variable predicates of where clause entries where CASE-TYPE = 2

For each IS-QUALIFY-LIST entry with

\[
\text{ISQ-RTNOL} = \text{CURR-RTNO and} \\
\text{ISQ-TYPE} = '2' \text{ and} \\
\text{ISQ-TYPE2-SOURCE} = 'E' \text{ or 'I' and} \\
\text{ISQ-EVAL-FLAG} = 1 \text{ and} \\
\text{ISQ-LEFT} = 'N' \text{ and} \\
\text{ISQ-MAP-ALG-IDL} = \text{blank}
\]

set ISQ-LEFT = 'Y'

write an RS4 access specification:

\[
\text{ACCESS-TYPE} = 'RS4', \\
\text{RS4-OP} = \text{ISQ-OP}, \\
\text{RS4-ISQ-PTR} = \text{ISQ-INDEX}, \\
\text{RS4-SIDE} = 'L', \\
\text{RS4-DFNO} = \text{ISQ-DFNOL}, \\
\text{RS4-DF-TYPE} = \text{ISQ-TYPEL}.
\]

5.2 Generate access specifications to check field-op-field predicates of where clause entries:

For each IS-QUALIFY-LIST entry with
Write an RSl access specification:

ACCESS-TYPE = 'RS1'
RS1-DFNOL = ISQ-DFNOL
RS1-DF-TYPEL = ISQ-TYPEL
RS1-DFNOR = ISQ-DFNOR
RS1-DF-TYPER = ISQ-TYPER
RS1-OP = ISQ-OP

5.2a Generate access specifications to transform field-op-variable entries to conceptual schema format if any predicate in the where clause participates in a complex mapping algorithm.

If CMA-FLAG = 'Y':

For each unique ISQ-DFIDL in the IS-QUALIFY-LIST entry with

ISQ-RTNOL = CURR-RTNO and
ISQ-TYPE = '2' and
ISQ-TYPE2-SOURCE = 'E' or 'I' and
ISQ-ALG-IDL = blank

Write an OUS access specification:

ACCESS-TYPE = 'OUS'
OUS-DFID = ISQ-DFIDL
OUS-DF-TYPE = ISQ-TYPEL
OUS-RTID = ISQ-RTIDL
OUS-DFNO = ISQ-DFNOR
OUS-TAGNO = CSQ-AUCL (ISQ-CSQ-PTR)

5.2b Generate access specifications to transform field-op-field entries to conceptual schema format if any predicate in the where clause participates in a complex mapping algorithm.
If CMA-FLAG = 'Y':

For each IS-QUALIFY-LIST entry with

\[
\begin{align*}
& \text{ISQ-SUBTRANS-IDL} = \text{SUB-ID} \text{ and} \\
& \text{ISQ-RTNOL} = \text{CURR-RTNO} \text{ and} \\
& \text{ISQ-TYPE} = '3' \text{ and} \\
& \text{ISQ-MAP-ALG-IDL} = \text{blank}
\end{align*}
\]

If IS-ACTION = 'D' or 'M'
write an OU5 access specification:

\[
\begin{align*}
& \text{ACCESS-TYPE} = 'OU5' \\
& \text{OU5-RTID} = \text{ISQ-RTIDL} \\
& \text{OU5-DF-TYPE} = \text{ISQ-TYPEL} \\
& \text{OU5-DFID} = \text{ISQ-DFIDL} \\
& \text{OU5-DFNO} = \text{ISQ-DFNOL} \\
& \text{OU5-TAGNO} = \text{CSQ-AUCL (ISQ-CSQ-PTR)}
\end{align*}
\]

If IS-ACTION = 'S' write an RF1 access specification

5.2c Generate access specifications to transform right sides of field-op-field where clause entries to conceptual schema format if any predicate participates in a complex mapping algorithm.

If CMA-FLAG = 'Y':

For each IS-QUALIFY-LIST entry with

\[
\begin{align*}
& \text{ISQ-SUBTRANS- IDR} = \text{SUB-ID} \text{ and} \\
& \text{ISQ-RTNOR} = \text{CURR-RTNO} \text{ and} \\
& \text{ISQ-TYPE} = '3' \text{ and} \\
& \text{ISQ-MAP-IDR} = \text{blank}
\end{align*}
\]

If IS-ACTION = 'D' or 'M'
write an OU5 access specification: generate MOVE D-dfno to TAG-tagno

\[
\begin{align*}
& \text{ACCESS-TYPE} = 'OU5' \\
& \text{OU5-RTNO} = \text{ISQ-RTNOR} \\
& \text{OU5-RTID} = \text{ISQ-RTIDR} \\
& \text{OU5-DFNO} = \text{ISQ-DFNOR} \\
& \text{OU5-DATATYPE} = \text{ISQ-TYPER} \\
& \text{OU5-TAGNO} = \text{CSQ-AUCR (ISQ-CSQ- PTR)}
\end{align*}
\]

If IS-ACTION = 'S'

16-16
Write an RFl access specification.

5.2d Generate access specifications to compare fields with fields from other records.

For each IS-QUALIFY-LIST entry with

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISQ-SUBTRANS-IDL</td>
<td>SUB-ID</td>
</tr>
<tr>
<td>ISQ-RTNOL</td>
<td>CURR-REC and</td>
</tr>
<tr>
<td>ISQ-TYPE</td>
<td>3' and</td>
</tr>
<tr>
<td>ISQ-LEFT</td>
<td>'N' and</td>
</tr>
<tr>
<td>ISQ-RTNOL not</td>
<td>ISQ-RTNOR and</td>
</tr>
<tr>
<td>ISQ-RIGHT</td>
<td>'Y' and</td>
</tr>
<tr>
<td>ISQ-ALG-IDL</td>
<td>blank</td>
</tr>
</tbody>
</table>

set ISQ-LEFT = 'Y'

write a RS4 access specification:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS-TYPE</td>
<td>'RS4'</td>
</tr>
<tr>
<td>RS4-DFNO</td>
<td>ISQ-DFNOL</td>
</tr>
<tr>
<td>RS4-OP</td>
<td>ISQ-OP</td>
</tr>
<tr>
<td>RS4-ISQ-PTR</td>
<td>ISQ-INDEX</td>
</tr>
<tr>
<td>RS4-SIDE</td>
<td>'L'</td>
</tr>
<tr>
<td>RS4-DF-TYPE</td>
<td>ISQ-TYPEL</td>
</tr>
</tbody>
</table>

5.2e Like Step 5.2d, but picking up fields from the right sides of predicates:

For each IS-QUALIFY-LIST entry with

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISQ-SUBTANS-IDR</td>
<td>SUB-ID</td>
</tr>
<tr>
<td>ISQ-RTNOR</td>
<td>CURR-REC and</td>
</tr>
<tr>
<td>ISQ-TYPE</td>
<td>3' and</td>
</tr>
<tr>
<td>ISQ-RIGHT</td>
<td>'N' and</td>
</tr>
<tr>
<td>ISQ-RTNOR not</td>
<td>ISQ-RTNOL and</td>
</tr>
<tr>
<td>ISQ-LEFT</td>
<td>'Y' and</td>
</tr>
<tr>
<td>ISQ-ALG-IDR</td>
<td>blank</td>
</tr>
</tbody>
</table>

set ISQ-RIGHT = 'Y'

write a RS4 access specification:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS-TYPE</td>
<td>'RS4'</td>
</tr>
<tr>
<td>RS4-DFNO</td>
<td>ISQ-DFNOR</td>
</tr>
<tr>
<td>RS4-OP</td>
<td>ISQ-OP</td>
</tr>
<tr>
<td>RS4-SIDE</td>
<td>'R'</td>
</tr>
<tr>
<td>RS4-ISQ-PTR</td>
<td>ISQ-INDEX</td>
</tr>
<tr>
<td>RS4-DF-TYPE</td>
<td>ISQ-TYPEL</td>
</tr>
</tbody>
</table>
5.3 Generate access specifications to output fields for retrieval actions:

5.3.1 If IS-ACTION not = 'S', '1', '2', or 'K', then go to Step 5.4.

5.3.1a For each IS-ACTION-LIST entry with

- IS-RTNO = CURR-REC and
- IS-FLAG = 'N' and
- IS-DF-DOESNT-REPEAT and
- IS-MAP-ALG-ID not = blank and
- IS-MAPPED-TO = 'Y'

set IS-FLAG = 1

write a RF3 access specification:

- ACCESS-TYPE = 'RF3'
- RF3-ALG-ID = IS-ALG-ID
- RF3-MOD-INST = CMA-MOD-INST
- RF3-PARM-NO = IS-PARM-NO
- RF3-IS-PTR = IS-INDEX

5.3.2 For each IS-ACTION-LIST entry with

- IS-RTNO = CURR-REC and
- IS-FLAG = 'N' and
- IS-DF-DOESNT-REPEAT and
- IS-ALG-ID = blank

set IS-FLAG = 1

write a RF1 access specification:

- ACCESS-TYPE = 'RF1'
- RF1-RTID = IS-RTID
- RF1-DFNO = IS-DFNO
- RF1-DFID = IS-DFID
- RF1-DF-TYPE = IS-DATA-TYPE
- RF1-IS-PTR = IS-INDEX

add IS-SIZE and IS-ND to NEXT-POSITION

5.3.3 Generate retrieval access specifications for repeating data fields by processing the OCCURS-TABLE.
Search the OCCURS-TABLE for OT-OCCURS-NEST entries where

\[
\begin{align*}
\text{OT-SUBTRANS} & = \text{current SUBTRANS-ID and} \\
\text{OT-MAPPED-TO} & = "Y" \text{ and} \\
\text{OT-RTNO} & = \text{current RTNO}
\end{align*}
\]

If no such entries are found, go to step 5.7.

Initialize the temporary working storage table TEMP-INDEX-STACK to zeros. Set TIS-INDEX to 1. Establish the current level of indexing as 1. Note: There are a maximum of 3 levels of indexing possible.

5.3.4 Determine if there are entries for the current level of indexing by checking the OT-INDEX-LEVELS field of the OT-OCCURS-NEST entries identified in step 5.3.3.

If no OT-OCCURS-NEST entry has an OT-INDEX-LEVELS greater than or equal to the current level of indexing, go to step 5.7.

For steps 5.3.4.1 through 5.3.4.4, consider only one OT-OCCURS-NEST entry from the set identified in step 5.3.3 which has an OT-INDEX-LEVELS greater than or equal to the current level of indexing.

5.3.4.1 Establish the DFNO of the index for the current level of indexing:

Set TIS-INDEX-DFNO = OT-DFNO
Increment TIS-USED.

5.3.4.2 Determine the initial value of the index.

1. If OT-INDEX-DFNO = 0

Write an OCI access specification to set the initial value of the index to 1:

\[
\begin{align*}
\text{ACCESS-TYPE} & = 'OCI' \\
\text{OCI-INDEX-DFNO} & = \text{TIS-INDEX-DFNO}
\end{align*}
\]
2. Else

Search the IS-QUALIFY-LIST for an entry where:

\[
\begin{align*}
\text{ISQ-TYPE} & = 2 \quad \text{and} \\
\text{ISQ-DFNOL} & = \text{OT-INDEX-DFNO} \\
\text{ISQ-DF-REPEAT-FLAG} & = 'I' \quad \text{and} \\
\text{ISQ-LEFT} & = 0 \\
\text{Set ISQ-FLAG} & = 1 \\
\text{If ISQ-RTIDL} & = \text{CURR-REC} \\
\end{align*}
\]

Write an OC1 access specification to set the initial value of the index to 1:

\[
\begin{align*}
\text{ACCESS-TYPE} & = 'OC1' \\
\text{OC1-INDEX-DFNO} & = \text{TIS-INDEX-DFNO} \\
\end{align*}
\]

Go to Step 5.3.4.3.

Else

Write an OC2 access specification to set the initial value of the index to a specific occurrence:

\[
\begin{align*}
\text{ACCESS-TYPE} & = 'OC2' \\
\text{OC2-INDEX-DFNO} & = \text{TIS-INDEX-DFNO} \\
\text{OC2-ISQ-PTR} & = \text{ISQ index} \\
\end{align*}
\]

Go to Step 5.3.4.4.

5.3.4.3 Determine the maximum value of the index.

Write an OC3 access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} & = 'OC3' \\
\text{OC3-INDEX-DFNO} & = \text{TIS-INDEX-DFNO} \\
\text{If OT-OCCURS-DEP-DFNO} & = 0 \\
\text{then} \\
\text{OC3-MAX-OCCURS} & = \text{OT-NUM-OCCURS} \\
\end{align*}
\]

16-20
OC3-OCCURS-DEP-DFNO = 0
If OT-OCCURS-DEP-DFNO not = 0
then
OC3-MAX-OCCURS = 0
OC3-OCCURS-DEP-DFNO = OT-OCCURS-DEP-DFNO

5.3.4.4 Generate the loop construct for this level of indexing.

Write an OC4 access specification:
ACCESS-TYPE = 'OC4'
OC4-INDEX-DFNO = TIS-INDEX-DFNO

5.3.5 For each OT-OCCURS-NEST entry identified in step 5.3.3, determine if the data field at the current level of indexing was selected for retrieval. Process as follows:

if OT-INDEX-LEVELS (OT-INDEX-1) not = current level of indexing, continue at step 5.3.5 with the next OT-OCCURS-NEST entry.

If all OT-OCCURS-NEST entries identified in step 5.3.3 have been processed, go to step 5.3.6.

Set OT-INDEX-2 = OT-STACK-USED (OT-INDEX-1)

Search the IS-ACTION-LIST for an entry where

IS-FLAG = 0 and
IS-RTNO = CURR-RTNO and
IS-DFNO = OT-DFNO

Set IS-FLAG = 1

Write an OC5 access specification:
ACCESS-TYPE = 'OC5'
OC5-DFNO = IS-DFNO
OC5-IS-PTR = IS-INDEX
OC5-IDX-DFNO1 = TIS-INDEX-DFNO (1)
OC5-IDX-DFNO2 = TIS-INDEX-DFNO (2)
OC5-IDX-DFNO3 = TIS-INDEX-DFNO (3)
OC5-NUM-INDEXES = OT-INDEX-LEVELS
(OT-INDEX-1)

5.3.6 Increment the current level of indexing, TIS-INDEX.

If current level of indexing > 3
   Go to step 5.7.
Else
   Go to step 5.3.4.

5.4 Generate access specifications to update fields for modify actions:

5.4.1 If IS-ACTION not = 'M', then go to Step 5.5.

5.4.2 Generate access specifications to convert update data values using complex mapping algorithms:

   For each COMPLEX-MAPPING-ALGORITHM-TABLE entry with
      CMA-SUBTRANSACTION = SUB-ID
      CMA-RETR-UPD = "U"

      5.4.2.1 Generate access specifications to move update data values to algorithm input parameters:

         For each CMA-PARAMETER-ENTRY with
            CMA-RTID = CURR-REC

         For each IS-ACTION-LIST entry with
            IS-RTID = CURR-REC and
            IS-FLAG = 0 and
            IS-MAPPED-TO-FLAG = 'Y' and
            IS-ALG-ID = CMA-MOD-ID:

            set IS-FLAG = 1

            write a FG3 access specification:

            ACCESS-TYPE = 'FG3'
            FG3-ALG-ID = IS-ALG-ID
            FG3-MOD-INST = CMA-MOD-INST
            FG3-PARM-NO = IS-PARM-NO
            FG3-IS-PTR = IS-INDEX

5.4.2.2 Generate access specifications to
move constant values to algorithm parameters.

For each CMA-PARAMETER-ENTRY with CMA-CONSTANT-VALUE filled in:

write a FG4 access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} &= 'FG4' \\
\text{FG4-CMA-CONSTANT} &= \text{CMA-CONST-VAL} \\
\text{FG4-ALG-ID} &= \text{CMA-MOD-ID} \\
\text{FG4-MOD-INST} &= \text{CMA-MOD-INST} \\
\text{FG4-PARM-NO} &= \text{CMA-PARM-NO}
\end{align*}
\]

5.4.2.3 Generate access specifications to call complex mapping algorithms.

Write a CAL access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} &= 'CAL' \\
\text{CAL-ALG-ID} &= \text{CMA-MOD-ID} \\
\text{CAL-PARM-COUNT} &= \text{CMA-PARM-COUNT} \\
\text{CAL-MOD-INST} &= \text{CMA-MOD-INST} \\
\text{CAL-MAP-DIR} &= \text{CMA-RETR-UPD}
\end{align*}
\]

5.4.2.4 Generate access specifications to move output algorithm parameters to entire records.

For each CMA-PARAMETER-ENTRY with CMA-RT-NO = CURR-RTNO and CMA-DF-NO not filled in:

write an FU2 access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} &= 'FU2' \\
\text{FU2-MOD-INST} &= \text{CMA-MOD-INST} \\
\text{FU2-ALG-ID} &= \text{CMA-MOD-ID} \\
\text{FU2-PARM-NO} &= \text{CMA-PARM-NO} \\
\text{FU2-RTID} &= \text{CURR-REC}
\end{align*}
\]

5.4.2.5 Generate access specifications to move output algorithm parameters to data fields.

For each CMA-PARAMETER-ENTRY with CMA-RT-NO = CURR-RTNO and CMA-DF-NO filled in:
write a FU access specification:

\begin{verbatim}
ACCESS-TYPE = 'FU1'
FU1-DFNO  = CMA-DF-NO
FU1-ALG-ID = CMA-MOD-ID
FU1-MOD-INST = CMA-MOD-INST
FU1-PARM-NO = COMA-PARM-NO
\end{verbatim}

5.4.3 Generate access specifications to move update data values to data fields.

For each IS-ACTION-LIST entry with

\begin{itemize}
  \item IS-RTID = CURR-REC and
  \item IS-FLAG = 0 and
  \item IS-MAP-ALG-ID = blank and
  \item IS-MAPPED-TO = 'Y'
\end{itemize}

set IS-FLAG = 1

Write an FU access specification:

\begin{verbatim}
ACCESS-TYPE = 'FU'
FUS-DFNO  = IS-DFNO
FUS-IS-PTR = IS-INDEX
FUS-NULL = 1 if IS-DELETE-ACTION or IS-NOT-MAPPED-TO = 0 if IS-MODIFY-ACTION or IS-MAPPED-TO
FUS-DF-TYPE = IS-DATA-TYPE
\end{verbatim}

5.4.4 If a FU access specification was written in Step 5.4.2, add an entry to the GROUP-TABLE for CURR-REC:

\begin{verbatim}
GR-RTID = CURR-REC
GR-RTNO = CURR-RTNO
GR-KEYFLAG = RK-KEY-CODE if RK-RTID = CURR-REC
GR-DELETE-FLAG = blank
GR-SETID = LAST-SETID-USED
GR-LOCK = IS-LOCK
\end{verbatim}

5.4.5 Go to Step 5.7.

5.5 Generate access specifications for delete actions:

5.5.1 If IS-ACTION not = D' then go to Step 5.6.
5.5.2 For the first IS-ACTION-LIST entry with IS-RTNO = CURR-RTNO and IS-FLAG = 0

Add an entry to the GROUP-TABLE:

- GR-RTID = CURR-REC
- GR-RTNO = CURR-RTNO
- GR-KEYFLAG = RK-KEY-CODE if RK-RTID = CURR-REC
  = blank if RK-RTID not = CURR-REC
- GR-SETID = LAST-SETID-USED
- GR-LOCK = IS-LOCK

5.5.2.1 If all IS-ACTION-LIST entries with IS-RTNO = CURR-RTNO have
IS-MAPPED-TO-FLAG = 'Y'

Set GR-DELETE-FLAG = 'RECORD'

5.5.2.2 Go to Step 5.7.

5.5.3 Delete mapped-to fields, retaining record if not entirely mapped to.

5.5.3a Generate access specifications to convert update data values using complex mapping algorithms:

Same as Step 5.4.2 except IS-LOCAL-VARIABLE is replaced with NULL-VALUE from DBMS on Host (E20) in Step 5.4.2.1.

5.5.3.1 For each IS-ACTION-LIST entry with
IS-RTNO = CURR-REC and
IS-FLAG = 'N' and
IS-MAPPED-TO-FLAG = 'Y' and
IS-MAP-ALG-ID = blank:

set IS-FLAG = 'Y'

write an FU access specification:

- ACCESS-TYPE = 'FU'
- FUS-DFNO = IS-DFNO
- FUS-IS-PTR = IS-INDEX
- FUS-NULL = 1 if
  IS-DELETE-ACTION or
  IS-NOT-MAPPED-TO
  = 0 if IS-MODIFY-ACTION
5.5.3.2 Set GR-DELETE-FLAG = 'FIELD'

5.5.4 Generate access specifications to move null values to repeating data fields.

Search the OCCURS-TABLE for OT-OCCURS-NEST entries where

OT-SUBTRANS = current SUBTRANS-ID and
OT-MAPPED-TO = "N" and
OT-RTNO = current RTNO

If no such entries are found, go to step 5.7.

Divide the OT-OCCURS-NEST entries into groups, based on OT-NESTID. All entries having the same OT-NESTID value belong to the same group.

Perform steps 5.5.5 thru 5.5.7 for each group of OT-OCCURS-NEST entries identified.

5.5.5 Determine if there are entries for the current level of indexing by checking the OT-INDEX-LEVELS field of the OT-OCCURS-NEST entries identified in step 5.5.4.

If no OT-OCCURS-NEST entry has an OT-INDEX-LEVELS greater than or equal to the current level of indexing, go to step 5.7.

For steps 5.5.5.1 through 5.5.5.4, consider only one OT-OCCURS-NEST entry from each group identified in step 5 which has an OT-INDEX-LEVELS greater than or equal to the current level of indexing.

5.5.5.1 Establish the DFNO of the index for the current level of indexing:

Set TIS-INDEX-DFNO = OT-DFNO.
Increment TIS-USED.

5.5.5.2 Determine the initial value of the index.

If OT-INDEX-DFNO = 0
Write an OC1 access specification to set the initial value of the index to 1:

ACCESS-TYPE = 'OC1'

16-26
OC1-INDEX-DFNO = TIS-INDEX-DFNO

5.5.5.3 Determine the maximum value of the index.

Write an OC3 access specification:

ACCESS-TYPE = 'OC3'
OC3-INDEX-DFNO = TIS-INDEX-DFNO

If OT-OCCURS-DEP-DFNO = 0
  OC3-MAX-OCURS = OT-NUM-OCCURS
  OC3-OCCURS-DEP-DFNO = 0

If OT-OCCURS-DEP-DFNO NOT = 0
  OC3-MAX-OCCURS = 0
  OC3-OCCURS-DEP-DFNO = OT-OCCURS-DEP-DFNO

5.5.5.4 Generate the loop construct for this level of indexing.

Write an OC4 access specification:

ACCESS-TYPE = 'OC4'
DC4-INDEX-DFNO = TIS-INDEX-DFNO

5.5.6 For each OT-OCCURS-NEST entry identified in step 5.5.4, determine if the data field at the current level of indexing was selected for retrieval. Process as follows:

If OT-INDEX-LEVELS (OT-INDEX-1) not = current level of indexing, continue at step 5.5.6 with the next OT-OCCURS-NEST entry.

If all OT-OCCURS-NEST entries identified in step 5.5.4 have been processed, go to step 5.5.7.

Set OT-INDEX-2 = OT-STACK-USED (OT-INDEX-1)

Search the IS-ACTION list for an entry where

  IS-FLAG = 0 and
  IS-RTNO = CURR-RTNO and
  IS-DFNO = OT-DFNO (TOT-INDEX-1, TOT-INDEX-2)

Set IS-FLAG = 1

Write an OC6 access specification:
ACCESS-TYPE = 'OC6'
OC6-DFNO = IS-DFNO
OC6-INDEX-DFNO1 = TIS-INDEX-DFNO (1)
OC6-INDEX-DFNO2 = TIS-INDEX-DFNO (2)
OC6-INDEX-DFNO3 = TIS-INDEX-DFNO (3)
OC6-NUM-INDEXES = OT-INDEX-LEVELS
                  (OT-INDEX-1)
OC6-DATATYPE = IS-DATATYPE (IS-INDEX)

5.5.7 Increment the current level of indexing, TIS-INDEX.

If current level of indexing > 3
    Go to step 5.7
Else
    Go to step 5.5.5

5.6 Generate access specifications to update fields for
insert actions:

If IS-ACTION not = 'I',
then generate an error message and abandon access
path.

5.6.1 Same as Step 5.5.2, setting GR-DELETE-FLAG =
blank.

5.6.1a Generate access specifications to convert
update data values using complex mapping
algorithms:

Same as Step 5.4.2 except that Step 5.4.2.1
is done for all IS-ACTION-List entries, not
just those with IS-MAPPED-TO-FLAG = 'Y', and
IS-LOCAL-VARIABLE is used only if
IS-MAPPED-TO-FLAG = 'Y', otherwise,
NULL-VALUE from DBMS on
Host (E20) is used.

5.6.2 For each IS-ACTION-List entry with
IS-RTNO = CURR-REC and
IS-FLAG = 'N' and
IS-MAP-ALG-ID = blank:

set IS-FLAG = 'Y'
write an FU access specification:

    ACCESS-TYPE = 'FU'
    FUS-DFNO = IS-DFNO
                  16-28
5.6.3 Generate access specifications to move null values to repeating data fields.

Search the OCCURS-TABLE for OT-OCCURS-NEST entries where

\[
\text{OT-SUBTRANS} = \text{current SUBTRANS-ID and} \\
\text{OT-MAPPED-TO} = "N" \text{ and} \\
\text{OT-RTNO} = \text{current RTNO}
\]

If no such entries are found, go to step 5.7.

Divide the OT-OCCURS-NEST entries into groups, based on OT-NESTID. All entries having the same OT-NESTID value belong to the same group.

Perform steps 5.6.4 thru 5.6.6 for each group of OT-OCCURS-NEST entries identified.

5.6.4 Determine if there are entries for the current level of indexing by checking the OT-INDEX-LEVELS field of the OT-OCCURS-NEST entries identified in step 5.6.3.

If no OT-OCCURS-NEST entry has an OT-INDEX-LEVELS greater than or equal to the current level of indexing, go to step 5.7.

For steps 5.6.4.1 through 5.6.4.4, consider only one OT-OCCURS-NEST entry from each group identified in step 5 which has an OT-INDEX-LEVELS greater than or equal to the current level of indexing.

5.6.4.1 Establish the DFNO of the index for the current level of indexing:

Set TIS-INDEX-DFNO = OT-DFNO.
Increment TIS-USED.

5.6.4.2 Determine the initial value of the index.

If OT-INDEX-DFNO = 0
Write an OC1 access specification to set the initial value of the index to 1:

ACCESS-TYPE = 'OC1'
OC1-INDEX-DFNO = TIS-INDEX-DFNO

5.6.4.3 Determine the maximum value of the index.

Write an OC3 access specification:

ACCESS-TYPE = 'OC3'
OC3-INDEX-DFNO = TIS-INDEX-DFNO

If OT-OCCURS-DEP-DFNO = 0
OC3-MAX-OCCURS = OT-NUM-OCCURS
OC3-OCCURS-DEP-DFNO = 0

If OT-OCCURS-DEP-DFNO NOT = 0
OC3-MAX-OCCURS = 0
OC3-OCCURS-DEP-DFNO = OT-OCCURS-DEP-DFNO

5.6.4.4 Generate the loop construct for this level of indexing.

Write an OC4 access specification:

ACCESS-TYPE = 'OC4'
OC4-INDEX-DFNO = TIS-INDEX-DFNO

5.6.5 For each OT-OCCURS-NEST entry identified in step 5.6.3, determine if the data field at the current level of indexing was selected for retrieval. Process as follows:

If OT-INDEX-LEVELS (OT-NDEX-l) not = current level of indexing, continue at step 5.6.5 with the next OT-OCCURS-NEST entry.

If all OT-OCCURS-NEST entries identified in step 5.6.3 have been processed, go to step 5.6.6.

Set OT-INDEX-2 = OT-STACK-USED (OT-INDEX-1)
Search the IS-ACTION list for an entry where
IS-FLAG = 0 and
IS-RTNO = CURR-RTNO and
IS-DFNO = OT-DFNO (TOT-INDEX-1, TOT-INDEX-2)

Set IS-FLAG = 1

Write an OC6 access specification:

ACCESS-TYPE = 'OC6'
OC6-DFNO = IS-DFNO
OC6-INDEX-DFNO1 = TIS-INDEX-DFNO (1)
OC6-INDEX-DFNO2 = TIS-INDEX-DFNO (2)
OC6-INDEX-DFNO3 = TIS-INDEX-DFNO (3)
OC6-NUM-INDEXES = OT-INDEX-LEVELS (TOT-INDEX-1)
OC6-DATATYPE = IS-DATATYPE (IS-INDEX)

5.6.6 Increment the current level of indexing, TIS-INDEX.

If current level of indexing > 3
   Go to step 5.7
Else
   Go to step 5.6.4

5.7 Generate access specifications to get fields for later comparison with fields from other records.

For each IS-QUALIFY-LIST entry with
   ISQ-RTNOL = CURR-REC and
   ISQ-TYPE = '3' and
   ISQ-LEFT = 'N' and
   ISQ-RTNOL not = ISQ-RTNOR and
   ISQ-RIGHT = 'N' and
   ISQ-MAP-ALG-IDL = blank

   set ISQ-LEFT = 'Y'

write an FGl access specification:

   ACCESS-TYPE = 'FG1'
   FG1-RTID = ISQ-RTIDL
   FG1-DFID = ISQ-DFIDL
   FG1-DFNO = ISQ-DFNOL
   FG1-ISQ-PTR = ISQ-INDEX
   FG1-SIDE = 'L'
   FG1-DF-TYPE = ISQ-TYPEL

5.8 Process in the same manner as Step 5.7, but pick up fields from the right sides of predicates:
For each IS-QUALIFY-LIST entry with
   ISQ-RTNOR = CURR-REC and
   ISQ-TYPE = '3' and
   ISQ-RIGHT = 'N' and
   ISQ-RTNOR not = ISQ-RTNOR and
   ISQ-LEFT = 'N' and
   ISQ-ALG-IDR = blank:
   set ISQ-RIGHT = 'Y'
   write an FG1 access specification:
      ACCESS-TYPE = 'FG1'
      FG1-RTID = ISQ-RTIDR
      FG1-DFID = ISQ-DFIDR
      FG1-DFNO = ISQ-DFNOR
      FG1-ISQ-PTR = ISQ-INDEX
      FG1-SIDE = 'R'
      FG1-DF-TYPE = ISQ-TYPE

6. Find the next step in the access path, looking upward.
Throughout, ST-MARK = 'Y' means that the SET-TABLE entry
has been accounted for in the access path.

6.1 Search the SET-TABLE for entries with ST-MARK = 'N'
and an ST-MEMBER(i) = CURR-REC.

   If there is none,
      go to Step 7
   else
      set LAST-SET-DOWN = blank.
      set LAST-SETID-USED = ST-SETID

6.2 Determine which SET-TABLE entries are value-based.

   6.2.1 Search the IS-ACTION-LIST for entries with
      IS-FLAG = 'N' and
      IS-RTNO = blank and
      an IS-RSNO = an ST-RSNO from Step 6.1.
      Record these IS-RSNOS.

   6.2.2 Search the IS-QUALIFY-LIST for entries with
      ISQ-LEFT = 'N' and
      ISQ-RTNOR = blank and
      an ISQ-RTNOL = an ST-RSNO from Step 6.1.
      Record these ISQ-RSNOLs.

   6.2.3 Search the IS-QUALIFY-LIST for entries with
      ISQ-RIGHT = 'N' and
      ISQ-RTNOR = blank and

16-32
an ISQ-RSNOR = an ST-RSNO from Step 6.1. Record these ISQ-RSNORs.

6.3 Process value-based sets.

For each RSNO represented in the set of qualifying entries from Step 6.2:

6.3.1 Write an SO1 access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} &= 'SO1' \\
\text{SS-SETID} &= \text{ST-SETID} \\
\text{SS-RTID} &= \text{ST-RTID}
\end{align*}
\]

Note that this command will result in changed run-unit currency only if the owner is found. Currency will be reset in Step 6.3.11.

6.3.2 Set ST-MARK = 'Y' for the SET-TABLE entry with ST-RSNO = RSNO.

6.3.3 Generate access specifications to compare the set-values with variables according to the where clause predicates:

For any IS-QUALIFY-LIST entry located in Step 6.2.2 because of its ISQ-RSNOL:

if ISQ-TYPE = 2,

generate an RS3 access specification:

\[
\begin{align*}
\text{ACCESS-TYPE} &= 'RS3' \\
\text{RS3-VALUE} &= \text{ISQ-STL-VALUE} \\
\text{RS3-OP} &= '=' \\
\text{RS3-RTID} &= \text{ISQ-RTIDL} \\
\text{RS3-ISQ-PTR} &= \text{ISQ-INDEX} \\
\text{RS3-SIDE} &= 'L'
\end{align*}
\]

concatenating the conditions from the qualifying entries to form a single IF statement,

set ISQ-LEFT = 'Y'.

6.3.4 Generate access specifications to either save the set-values for later comparison with fields according to where clause predicates, or to do the comparisons:
For any IS-QUALIFY-LIST entry located in Step 6.2.2 because of its ISQ-RSNOL:

if ISQ-TYPE = 3 and ISQ-RIGHT = 'N'
genrate an FG2 access specification:

   ACCESS-TYPE = 'FG2'
   FG2-VALUE   = ISQ-STL-VALUE
   FG2-ISA-PTR = ISQ-INDEX
   FG2-SIDE    = 'L'

set ISQ-LEFT = 'Y'.

if ISQ-TYPE = 3 and ISQ-RIGHT = 'Y'
genrate an RS3 access specification:

   ACCESS-TYPE = 'RS3'
   RS3-VALUE   = ISQ-STL-VALUE
   RS3-OP      = '='
   RS3-RTID    = ISQ-ISQ-RTIDL
   RS3-ISQ-PTR = ISQ-INDEX
   RS3-SIDE    = 'L'

set ISQ-LEFT = 'Y'.

6.3.5 Same as Step 6.3.4, except for the right-side set values:

For any IS-QUALIFY-LIST entry located in Step 6.2.3 because of its ISQ-RSNOR:

if ISQ-TYPE = 3 and ISQ-LEFT = 'N'
genrate an FG2 access specification:

   ACCESS-TYPE = 'FG2'
   FG2-VALUE   = ISQ-STR-VALUE
   FG2-ISQ-PTR = ISQ-INDEX
   FG2-SIDE    = 'R'

set ISQ-RIGHT = 'Y'.

if ISQ-TYPE = 3 and
ISQ-LEFT = 'Y'

generate an RS3 access specification:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS-TYPE</td>
<td>'RS3'</td>
</tr>
<tr>
<td>RS3-OP</td>
<td>'='</td>
</tr>
<tr>
<td>RS3-VALUE</td>
<td>ISQ-STK-VALUE</td>
</tr>
<tr>
<td>RS3-RTID</td>
<td>ISQ-RTIDR</td>
</tr>
<tr>
<td>RS3-ISQ-PTR</td>
<td>ISQ-INDEX</td>
</tr>
<tr>
<td>RS3-SIDE</td>
<td>'R'</td>
</tr>
</tbody>
</table>

set ISQ-RIGHT = 'Y'.

6.3.6 If IS-ACTION not = 'S', '1', '2', or 'K', then go to Step 6.3.7.

For each IS-ACTION-LIST entry located in Step 6.2.1 because of its IS-RSNO:

generate an RF2 access specification, to pick up the set value:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS-TYPE</td>
<td>'RF2'</td>
</tr>
<tr>
<td>RF2-VALUE</td>
<td>IS-ST-VALUE</td>
</tr>
<tr>
<td>RF2-IS-PTR</td>
<td>IS-INDEX</td>
</tr>
</tbody>
</table>

set IS-FLAG = 'Y'.

add IS-SIZE and IS-ND to NEXT-POSITION

Go to Step 6.3.11.

6.3.7 If IS-ACTION not = 'M' then go to Step 6.3.8.

Generate code for the following logic:

If the set entry's value = IS-LOCAL-VARIABLE
    insert into that set
else if already a member in that set disconnect from the set.
For each IS-ACTION-LIST entry located in Step 6.2.1 because of its IS-RSNOS:

generate an IT2 access specification:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS-TYPE</td>
<td>'IT2'</td>
</tr>
<tr>
<td>IT2-VALUE</td>
<td>IS-ST-VALUE</td>
</tr>
<tr>
<td>IT2-OP</td>
<td>'='</td>
</tr>
<tr>
<td>IT2-IS-PTR</td>
<td>IS-INDEX</td>
</tr>
<tr>
<td>IT2-ISQ-PTR</td>
<td>0</td>
</tr>
</tbody>
</table>
generate an SI access specification:
    ACCESS-TYPE = 'SI'
    SI-SETID = ST-SETID
    SI-RTID = ST-RTID

generate an IE access specification:
    ACCESS-TYPE = 'IE'

generate an SO1 access specification:
    ACCESS-TYPE = 'SO1'
    SS-SETID = ST-SETID
    SS-RTID = ST-RTID

generate an SD access specification:
    ACCESS-TYPE = 'SD'
    SD-SETID = ST-SETID
    SD-RTID = ST-RTID

generate an EI access specification:
    ACCESS-TYPE = 'EI'

generate another EI access specification:
    ACCESS-TYPE = 'EI'

set IS-FLAG = 'Y'.

Go to Step 6.3.11.

6.3.8 If IS-ACTION not = 'D'
go to Step 6.3.9.

Generate code for the following logic:
    If a member in the set,
disconnect it.
    For each IS-ACTION-LIST entry located in
Step 6.2.1 because of its IS-RSNOS:

generate an SO1 access specification:
    ACCESS-TYPE = 'SO1'
    SS-SETID = ST-SETID
    SS-RTID = ST-RTID

generate an SD access specification:
    ACCESS-TYPE = 'SD'
    SD-SETID = ST-SETID
    SD-RTID = ST-RTID

generate an EI access specification:
    ACCESS-TYPE = 'EI'

16-36
set IS-FLAG = 'Y'.
Go to Step 6.3.11.

6.3.9 If IS-ACTION not = 'I',
issue an error message.

Generate code for the following logic:
If the set entry's value = IS-LOCAL-VARIABLE
insert into that set.
For each IS-ACTION-LIST entry located in Step 6.2 because of its IS-RSNOs:

generate an IT2 access specification:
ACCESS-TYPE = 'IT2'
IT2-VALUE = IS-ST-VALUE
IT2-OP = '='
IT2-IS-PTR = IS-INDEX
IT2-ISQ-PTR = 0

generate an SI access specification:
ACCESS-TYPE = 'SI'
SI-SETID = ST-SETID
SI-RTID = ST-RTID

generate an EI access specification:
ACCESS-TYPE = 'EI'
Set IS-FLAG = 'Y'

6.3.10 (This Step was removed.)

6.3.11 Close the open conditional from Step 6.3.1
for this record set and reset the currency
by doing the following:

Write an EI access specification:
ACCESS-TYPE = 'EI'
Write an RC access specification:
ACCESS-TYPE = 'RC'
RCS-RTID = CURR-REC

6.4 Process relation-class-based sets.

Find any entries that specify traversal of the
identified record sets and change of currency, by
doing the following:

16-37
Search the SET-TABLE for an entry with
ST-MEMBER(1) = CURR-REC and
ST-MARK = 'N'.

If one is found, then:

6.4.1 Set ST-MARK = 'Y'

6.4.2 Write an SO2 access specification:
   ACCESS-TYPE = 'SO2'
   SS-SETID = ST-SETID
   SS-RTID = ST-RTID

   Note that this command will result in changed
run-unit currency if the owner is found. The
currency will not be reset.

6.4.3 Push CURR-REC onto the RTID-STACK, set CURR-REC = ST-OWNER, from the SET-TABLE entry
   with ST-RSNO = IS-RSNO from Step 6.4.

6.5 (This Step was changed to Step 6.4.4.)

6.6 Go to Step 5.

7. Return one level downward in the path, by doing the
following:

7.1 If the RTID-STACK is empty (i.e. if the path has not
gone upward) go to Step 8.

7.2 Pop the RTID-STACK into CURR-REC
   Write an RC access specification:
   ACCESS-TYPE = 'RC'
   RCS-RTID = CURR-REC

7.3 Go to Step 6.

8. The access path has now been constructed upwards from the
candidate port record and we can look downward.

8.1 Search the SET-TABLE for entries with
    ST-MARK = 'N' and
    ST-OWNER = CURR-REC
   If there is none, go to Step 9.

8.2 Search the IS-ACTION-LIST for entries with
    IS-FLAG = 'N' and
IS-RTNO = blank and
an IS-RSNO(i) = an ST-RSNO from Step 8.1
Record these RSNOs.

Search the IS-QUALIFY-LIST for entries with
ISQ-LEFT = 'N' and
ISQ-RTNOL = blank and
an ISQ-RSNOL(i) = an ST-RSNO from Step 8.1
Record these RSNOs.

Search the IS-QUALIFY-LIST for entries with
ISQ-RIGHT = 'N' and
ISQ-RTNOR = blank and
an ISQ-RSNOR(i) = an ST-RSNO from Step 8.1
Record these RSNOs.

8.3 For each RSNO in the set recorded in Step 8.2:

8.3.1 If ST-TOTAL-NUM-MEMBERS = ST-NUM-MEMBERS in the
SET-TABLE entry with ST-RSNO = RSNO for
this iteration through Step 8.3

write an SMI access specification:
ACCESS-TYPE = 'SMI'
SS-SETID = ST-SETID
SS-RTID = ST-RTID

else write an SMI access specification:
ACCESS-TYPE = 'SMI'
SS-SETID = ST-SETID
SS-RTID = ST-RTID

Note that ST-NUM-MEMBERS = ST-TOTAL-NUM-MEMBERS
or one. The SMI access specification will
result in changed run-unit currency if at least
one member is found. Currency will be reset in
Step 8.3.8.

8.3.2 through 8.3.11

Perform Steps 6.3.2 through 6.3.11, replacing
all references to:

<table>
<thead>
<tr>
<th>Step 6.2</th>
<th>by Step 8.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.1</td>
<td>8.2.1</td>
</tr>
<tr>
<td>6.2.2</td>
<td>8.2.2</td>
</tr>
<tr>
<td>6.2.3</td>
<td>8.2.3</td>
</tr>
<tr>
<td>6.3</td>
<td>8.3</td>
</tr>
<tr>
<td>6.3.1</td>
<td>8.3.1</td>
</tr>
</tbody>
</table>

16-39
8.4 Search for entries that specify set traversal and change of currency, by doing the following:

For each SET-TABLE entry found in Step 8.1:

8.4.a Determine whether the set needs to be traversed.

Find all the IS-QUALIFY-LIST entries with either

| ISQ-RTNOL  | = CURR-REC and |
| ISQ-TYPE   | = '3' and |
| ISQ-RTNOR  | = any ST-MEMBER in any of |
|            | the SET-TABLE entries |
|            | from Step 8.1 and |
| ISQ-RIGHT  | = 'N' |

or

| ISQ-RTNOR  | = CURR-REC and |
| ISQ-TYPE   | = '3' and |
| ISQ-RTNOL  | = any ST-MEMBER in any of |
|            | the SET-TABLE entries |
|            | from Step 8.1 and |
| ISQ-LEFT   | = 'N'. |

If any such IS-QUALIFY-LIST entries are found, proceed to Step 8.4.1 with this SET-TABLE entry. If no such IS-QUALIFY-LIST entries are found, continue with Step 8.4.a for the next SET-TABLE entry.

8.4.1 Set ST-MARK = 'Y' for the SET-TABLE entry with ST-RSNO = ISQ-ISNOL(1).

8.4.2 If ST-TOTAL-NUM-MEMBERS = ST-NUM-MEMBERS

write an SM2 access specification:

| ACCESS-TYPE | = 'SM2' |
| SS-SETID    | = ST-SETID |
| SS-RTID     | = ST-RTID |

16-40
else write an SM2 access specification:
   ACCESS-TYPE = 'SM2'
   SS-SETID    = ST-SETID
   SS-RTID     = ST-RTID

Note that ST-NUM-MEMBERS = ST-TOTAL-NUM-MEMBERS
or one. This command will change run-unit
currency if at least one member is found.
Currency will not be reset.

8.4.3 Set LAST-SET-DOWN = SS-SETID.
     Set LAST-SET-USED = SS-SETID.

8.4.4 Same as Step 8.3.12.

8.5 If an SM access specification was written in Step 8.4
     set CURR-REC = ST-MEMBER(1).
     Go to Step 5.

9. Generate access specifications to perform the modify,
    insert or delete actions, or to write selected results to an
    output file.

9a.1 If CASE-TYPE = 6
      Write a 1IF access specification:
         ACCESS-TYPE = '1IF'

9a.2 If IS-ACTION = 'D' or 'M' and CMA-SWITCH = 'Y'
      Write a 'CIF' access specification:
         ACCESS-TYPE = 'CIF'

9a.3 If IS-ACTION = 'S', '1', '2', or 'K'
      Write a PIO access specification:
         ACCESS-TYPE = 'PIO'

9a.4 ELSE
     For each entry in the GROUP-TABLE

9a.4.1 Write a RC access specification:
         ACCESS-TYPE = 'RC'
         RCS-RTID    = CURR-REC

9a.4.2 Write an MR2 access specification:
         ACCESS-TYPE = 'MR2'
         MR2-RTNO    = GR-RTNO
         MR2-RTID    = GR-RTID

9a.4.3 If IS-ACTION = 'M'
         16-41
If GR-KEYFLAG = 1
Write a RUK access specification:
  ACCESS-TYPE = 'RUK'
  REC-SELECT-SPEC-PTR = RK-INDEX

Else
Write a RU2 access specification:
  ACCESS-TYPE = 'RU2'
  RU2-RTID = GR-RTID

9a.4.4 If IS-ACTION = 'D'
If GR-DELETE-FLAG = 'FIELD'
  Process same as 9a.4.3
Else
If GR-KEYFLAG = 'U' or 'D'
  Write a RDK access specification.
    REC-SELECT-SPEC-PTR = RK-INDEX
Else
  Write a RD2 access specification:
    ACCESS-TYPE = 'RD2'
    RD2-RTID = GR-RTID
    RD2-SETID = GR-SETID

9a.4.5 If IS-ACTION = 'I'
If GR-KEYFLAG = 'U' or 'D'
  Write a RIK access specification:
    ACCESS-TYPE = 'RIK'
    REC-SELECT-SPEC-PTR = RK-INDEX
Else
  Write a R12 access specification:
    ACCESS-TYPE = 'R12'
    R12-RTID = GR-RTID

9a.5 Write an EP access specification:
  ACCESS-TYPE = 'EP'

10. Return to PRE13 to have PRE7 invoked.

Constraints

Note that this algorithm requires that if a CS AUC maps to more than one IS record set, then those record sets must all have the same owner record type and the same member record types. Not being a participant in any of the mapped-to record sets cannot map to an AUC value.

If any conditions in the IS-QUALIFY-LIST for this subtransaction participate in complex mapping algorithms, the entire NDML WHERE clause must be evaluated at the conceptual schema level.
16.3 Outputs

* ACCESS PATH TABLE
* CONTAINS THE ACCESS PATH FOR ONE SUBTRANSACTION
* FOR AN SDLN REQUEST.

01 ACCESS-PATHS.
  03 AT-MAX PIC 999 VALUE 200.
  03 AT-USED PIC 999.
  03 ACCESS-TYPE-ENTRY OCCURS 200 INDEXED BY AT-INDEX.
     05 ACCESS-TYPE-CODE PIC XXX.
        88 CAL-TYPE VALUE "CAL".
        88 CIF-TYPE VALUE "CIF".
        88 EI-TYPE VALUE "EI".
        88 EP-TYPE VALUE "EP".
        88 FG1-TYPE VALUE "FG1".
        88 FG2-TYPE VALUE "FG2".
        88 FG3-TYPE VALUE "FG3".
        88 FG4-TYPE VALUE "FG4".
        88 FU-TYPE VALUE "FU".
        88 FU1-TYPE VALUE "FU1".
        88 FU2-TYPE VALUE "FU2".
        88 FU3-TYPE VALUE "FU3".
        88 FU4-TYPE VALUE "FU4".
        88 IE-TYPE VALUE "IE".
        88 IIF-TYPE VALUE "IIF".
        88 IT2-TYPE VALUE "IT2".
        88 MR1-TYPE VALUE "MR1".
        88 MR2-TYPE VALUE "MR2".
        88 N:5-TYPE VALUE "N:5".
        88 NXS-TYPE VALUE "NXS".
        88 OC1-TYPE VALUE "OC1".
        88 OC2-TYPE VALUE "OC2".
        88 OC3-TYPE VALUE "OC3".
        88 OC4-TYPE VALUE "OC4".
        88 OC5-TYPE VALUE "OC5".
        88 OC6-TYPE VALUE "OC6".
        88 OU4-TYPE VALUE "OU4".
        88 OC5-TYPE VALUE "OC5".
        88 PIO-TYPE VALUE "PIO".
        88 RA-TYPE VALUE "RA".
        88 RAI-TYPE VALUE "RAI".
        88 RC-TYPE VALUE "RC".
        88 RDK-TYPE VALUE "RDK".
        88 RD2-TYPE VALUE "RD2".
**8 RF1-TYPE** VALUE "RF1".
**8 RF2-TYPE** VALUE "RF2".
**8 RF3-TYPE** VALUE "RF3".
**8 RIK-TYPE** VALUE "RIK".
**8 RI2-TYPE** VALUE "RI2".
**8 RK-TYPE** VALUE "RK".
**8 RK1-TYPE** VALUE "RK1".
**8 RK2-TYPE** VALUE "RK2".
**8 RK3-TYPE** VALUE "RK3".
**8 RS1-TYPE** VALUE "RS1".
**8 RS3-TYPE** VALUE "RS3".
**8 RS4-TYPE** VALUE "RS4".
**8 RS5-TYPE** VALUE "RS5".
**8 RUK-TYPE** VALUE "RUK".
**8 RU2-TYPE** VALUE "RU2".
**8 SD-TYPE** VALUE "SD".
**8 SI-TYPE** VALUE "SI".
**8 SM1-TYPE** VALUE "SM1".
**8 SM2-TYPE** VALUE "SM2".
**8 SO1-TYPE** VALUE "SO1".
**8 SO2-TYPE** VALUE "SO2".
**8 UIF-TYPE** VALUE "UIF".

05 REC-SELECT-SPEC-PTR PIC 999.
* ACCESS PATH INFORMATION TABLE

* THIS IS A COLLECTION OF INFORMATION STORED IN A NUMBER OF VARIOUS TABLES USED BY THE ACCESS PATH TABLE AND THE GENERIC CODASYL TABLE. SEE CDMP SPEC, PRE6

* APINFO.INC

  01 AP-INFO-TABLE.
  02 API-MAX       PIC 9(3)     VALUE 200.
  02 API-USED     PIC 9(3) .
  02 API-ALL-TABLES-DEF OCCURS 200 TIMES INDEXED BY API-INDEX.
  03 API-DEF.
      05 FILLER     PIC X(112).

* REL 2.3 Complex Mapping algorithm call

  03 CAL-SPEC REDEFINES API-DEF.
      05 CAL-ALG-ID  PIC X(8).
      05 CAL-MOD-INST PIC 999.
      05 CAL-PARM-COUNT PIC 999.

* Old: Move data field to ISQ variable

  03 FG1-SPEC REDEFINES API-DEF.
      05 FG1-RTID      PIC X(30).
      05 FG1-DFNO      PIC 9(6).
      05 FG1-DFID      PIC X(30).
      05 FG1-ISQ-PTR   PIC 999.
      05 FG1-SIDE      PIC X.
      05 FG1-DF-TYPE   PIC X.

* Old: Move set value to ISQ variable

  03 FG2-SPEC REDEFINES API-DEF.
      05 FG2-VALUE     PIC X(30).
      05 FG2-ISQ-PTR   PIC 999.
      05 FG2-SIDE      PIC X.

* Old: Move runtime var/value to input CMA parameter

  03 FG3-SPEC REDEFINES API-DEF.
      05 FG3-ALG-ID    PIC X(3).

16-46
05 FG3-MOD-INST  PIC 999.
05 FG3-PARM-NO   PIC 999.
05 FG3-IS-PTR    PIC 999.

* Rel 2.3: Move constant to CMA parameter

03 FG4-SPEC REDEFINES API-DEF.
  05 FG4-ALG-ID    PIC X(8).
  05 FG4-MOD-INST  PIC 999.
  05 FG4-PARM-NO   PIC 999.
  05 FG4-CONSTANT  PIC X(30).

* Old: Move update value or null to data field

03 FUS-SPEC REDEFINES API-DEF.
  05 FUS-DFNO      PIC 9(6).
  05 FUS-IS-PTR    PIC 999.
  05 FUS-NNULL     PIC X.
  05 FUS-DF-TYPE   PIC X.

* Rel 2.3: Move output CMA parameter to data field

03 FU1-SPEC REDEFINES API-DEF.
  05 FU1-DFNO      PIC 9(6).
  05 FU1-ALG-ID    PIC X(8).
  05 FU1-MOD-INST  PIC 999.
  05 FU1-PARM-NO   PIC 999.

* Rel 2.3: Move output CMA parameter to record

03 FU2-SPEC REDEFINES API-DEF.
  05 FU2-RTID      PIC X(30).
  05 FU2-ALG-ID    PIC X(8).
  05 FU2-MOD-INST  PIC 999.
  05 FU2-PARM-NO   PIC 999.

* Rel 2.3: Move data field to input CMA parameter

03 FU3-SPEC REDEFINES API-DEF.
  05 FU3-RTID      PIC X(30).
  05 FU3-DFNO      PIC 9(6).
  05 FU3-DPID      PIC X(30).
  05 FU3-DF-TYPE   PIC X.
  05 FU3-IS-PTR    PIC 999.
  05 FU3-ALG-ID    PIC X(8).
  05 FU3-MOD-INST  PIC 999.
  05 FU3-PARM-NO   PIC 999.

* Rel 2.3: Move record to input CMA parameter
03 FU4-SPEC REDEFINES API-DEF.
  05 FU4-RTID PIC X(30).
  05 FU4-ALG-ID PIC X(8).
  05 FU4-MOD-INST PIC 999.
  05 FU4-PARM-NO PIC 999.

* Old: If set-value op ISQ variable

03 IT2-SPEC REDEFINES API-DEF.
  05 IT2-OP PIC XX.
  05 IT2-VALUE PIC X(30).
  05 IT2-ISQ- PTR PIC 999.
  05 IT2-IS- PTR PIC 999.

* Rel 2.3: Move record from schema to ws and vice versa

03 MR-SPEC REDEFINES API-DEF.
  05 MR-RTHO PIC 9(6).
  05 MR-RTID PIC X(30).

* Rel 2.3: Move runtime value to ISQL variable

03 MVS-SPEC REDEFINES API-DEF.
  05 MVS-ISQ-PTR PIC 999.

* Rel 2.3: Set the index data field to a value of 1.

03 OC1-SPEC REDEFINES API-DEF.
  05 OC1-INDEX-DFNO PIC 9(6).

* Rel 2.3: Move variable containing the number of
  occurrences or occurs depending on value to
  a local variable.

03 OC2-SPEC REDEFINES API-DEF.
  05 OC2-INDEX-DFNO PIC 9(6).
  05 OC2-ISQ-PTR PIC 999.

* Rel 2.3: Move data field or value containing the number
  of occurrences or occurs depending on value to
  local variable.

03 OC3-SPEC REDEFINES API-DEF.
  05 OC3-INDEX-DFNO PIC 9(6).
  05 OC3-OCCURS-DEP-DFNO PIC 9(6).
  05 OC3-MAX-OCCURS PIC 99.

* Rel 2.3: Determine if the current index is greater than

16-43
* the maximum index value.

03 0C4-SPEC REDEFINES API-DEF.
  05 0C4-INDEX-DFNO PIC 9(6).

* Rel 2.3: Move an indexed field to the results record.

03 0C5-SPEC REDEFINES API-DEF.
  05 0C5-INDEX-DFNO PIC 9(6).

* Rel 2.3: Move a null value to an indexed field

03 0C6-SPEC REDEFINES API-DEF.
  05 0C6-INDEX-DFNO PIC 9(6).

* Rel 2.3: Move CMA output parameter to tag.

03 OU4-SPEC REDEFINES API-DEF.
  05 OU4-ALG-ID PIC X(8).

* Rel 2.3: Move retrieved data field to tag.

03 OU5-SPEC REDEFINES API-DEF.
  05 OU5-RTID PIC X(30).

* Old: Area sweep access path

03 RA-SPEC REDEFINES API-DEF.
  05 RAS-RTID PIC X(30).
  05 RAS-AREAID PIC X(30).
* Old: Reset currency

03 RC-SPEC REDEFINES API-DEF.
   05 RCS-RTID PIC X(30).

* Old: Delete next record

03 RD2-SPEC REDEFINES API-DEF.
   05 RD2-RTID PIC X(30).
   05 RD2-SETID PIC X(30).

* Old: Move field to result rec

03 RF1-SPEC REDEFINES API-DEF.
   05 RF1-RTID PIC X(30).
   05 RF1-DFNO PIC 9(6).
   05 RF1-DFID PIC X(30).
   05 RF1-DF-TYPE PIC X.
   05 RF1-IS-PTR PIC 999.

* Old: Move value to result rec

03 RF2-SPEC REDEFINES API-DEF.
   05 RF2-VALUE PIC X(30).
   05 RF2-IS-PTR PIC 999.

* Rel 2.3: Move CMA parameter to result rec

03 RF3-SPEC REDEFINES API-DEF.
   05 RF3-ALG-ID PIC X(8).
   05 RF3-MOD-INST PIC 999.
   05 RF3-PARM-NO PIC 999.
   05 RF3-IS-PTR PIC 999.

* Old: Insert next record

03 R12-SPEC REDEFINES API-DEF.
   05 R12-RTID PIC X(30).

* Rel 2.3: Start loop for multiple values of key

03 RK1-SPEC REDEFINES API-DEF.
   05 RK1-LOOP-MAX PIC 99.

* Rel 2.3: Move nth value to key

03 RK2-SPEC REDEFINES API-DEF.
   05 RK2-RTID PIC X(30).
   05 RK2-RK-INDEX PIC 999.

16-50
05 RK2-LOOP-COUNT PIC 99.
05 RK2-DFID PIC X(30).

* Old: If not dfid-left op dfid-right

03 RS1-SPEC REDEFINES API-DEF.
  05 RS1-DFNOL PIC 9(6).
  05 RS1-DF-TYPEL PIC X.
  05 RS1-OP PIC XX.
  05 RS1-DFNOR PIC 9(6).
  05 RS1-DF-TYPER PIC X.

* Old: If not value op variable

03 RS3-SPEC REDEFINES API-DEF.
  05 RS3-RTID PIC X(30).
  05 RS3-VALUE PIC X(30).
  05 RS3-OP PIC XX.
  05 RS3-ISQ-PTR PIC 999.
  05 RS3-SIDE PIC X.

* Old: If not dfid op ISQ-variable

03 RS4-SPEC REDEFINES API-DEF.
  05 RS4-OP PIC XX.
  05 RS4-DFNO PIC 9(6).
  05 RS4-ISQ-PTR PIC 999.
  05 RS4-SIDE PIC X.
  05 RS4-DF-TYPE PIC X.

* Rel 2.3: If check for ORed conditions in same record.

03 RS5-SPEC REDEFINES API-DEF.
  05 RS5-DFNO PIC 9(6).
  05 RS5-OP PIC XX.
  05 RS5-ISQ-PTR PIC 999.
  05 RS5-IF-OR PIC XX.
  05 RS5-SIDE PIC X.
  05 RS5-DF-TYPE PIC X.

* Old: Update next record

03 RU2-SPEC REDEFINES API-DEF.
  05 RU2-RTID PIC X(30).

* Old: Handles SM1, SM2, SO1, SO2, SD, SI

03 SET-SPEC REDEFINES API-DEF.

16-51
05 SS-RTID PIC X(30).
05 SS-SETID PIC X(30).

* REL 2.3: Handles Union Discriminator

03 UIF-SPEC REDEFINES API-DEF.
05 UIF-RTNO PIC 9(6).
16.4 Internal Requirements

1. A temporary stack to hold the index DFNOs for 3 levels of repeating fields.

   01 TEMP-INDEX-STACK.
   03 TIS-MAX          PIC 99.
   03 TIS-USED         PIC 99.
   03 TIS-ENTRY        OCCURS 3 TIMES
                       INDEXED BY TIS-INDEX.
   05 TIS-INDEX-DFNO   PIC 9(6).

2. Table used first to hold all the unique record types associated with the subtransaction; used to hold the record types which must be updated, deleted or inserted at the end of each iteration of the access path.

   01 GROUP-TABLE.
   03 GR-MAX          PIC 99.
   03 GR-USED         PIC 99.
   03 GR-ENTRY        OCCURS 50 TIMES
                       INDEXED BY GR-INDEX.
   05 GR-RTID         PIC X(30).
   05 GR-TRNO         PIC 9(6).
   05 GR-KEYFLAG      PIC X(6).
   05 GR-SETID        PIC X(30).
   05 GR-LOCK         PIC .

3. Internal switches and variables:

   CURR-REC            = RTID of current record type
   NEXT-POSITION       = temporary pointer to next open position in buffer
   LAST-SET-DOWN       = setid for last set type traversed downward
   LAST-SET-USED       = setid of set type being traversed
   CURR-RTNO           = RTNO of current record type
SECTION 17
FUNCTION PRE7 - TRANSFORM IS ACCESS PATH/GENERIC DML

The IS Access Path/Generic DML Transformer is invoked at precompile-time. It transforms IS Access Path specifications produced by PRE6 - Select IS Access Path into proper code structures to traverse the local databases. The generic DML will later be transformed to the DML of a particular DBMS (e.g. TOTAL, IMS, IDMS) by the Generic/Specific DML Transformers of the Request Process Generators, PRE9. PRE7 is called by PRE5.

The IS Access Path/Generic DML Transformer builds DML code in the form of nested "C-structures." A stack is employed to retain the bottoms of the C-structures.

17.1 Inputs

An IS Access Path through a single local database generated by function PRE6 - Select IS Access Path. This input is the structure ACCESS-PATH and the accompanying tables specified as outputs in the PRE6 development specification, including the RECORD-KEY-TABLE.

17.2 Processing

1. If NDML-NO = 1, then set the loop labeler: i = 0.
2. Transform the port specification.
   2.1 If ACCESS-TYPE not = 'RK', then go to Step 2.5a.
   2.2 Set RK-INDEX = REC-SELECT-SPEC-PTR.
   2.3 If RK-KEYCODE (RK-INDEX) not = 'U', then go to Step 2.4. Generate DML for an unique primary key for which a single value was specified:
      2.3.1 Find record key components in the RECORD-KEY-TABLE:
      For j = 1 to RK-DF-USED (RK-INDEX):
      Set RS2-DFID(j) = RK-DFID(RK-INDEX, j)
      RS2-VARIABLE = ISQL-VAR-n where n = RK-ISQ-PTR or RK-IS-PTR
      2.3.2 Generate DML:
2.3.3 Push onto DML stack:

ENDLOOP.i

2.3.4 Go to Step 3.

2.4 If RK-KEYCODE (RK-INDEX) not = 'D', then generate an error message and abandon the access path.

Generate DML for a duplicate key for which a single value was specified:

2.4.1 Increment the loop labeler: \( i = i + 1 \)

2.4.2 Find record key components in the RECORD-KEY-TABLE:

- Set \( RS2-DFID(j) = RK-DFID(K-INDEX),j \)
- \( RS2-VARIABLE = ISQL-VAR-n \) where \( n = RI-ISQ-PTR \) or \( RK-IS-PTR \)

2.4.3 Generate DML:

```
LOOP.i-1
FFR rs2-rtid,
    {rs2-dfid(j)=rs2-variable(j)},
    rs2-lock
IFFRECNOTFOUND
EXITLOOP.i
ENDIF
```

2.4.4 Push onto DML stack:

```
ENDLOOP.i
FNR rs2-rtid,
    {rs2-dfid(j)=rs2-variable(j)},
    rs2-lock
```

Note that the FNR is now on top of the stack.
2.4.5 Go to Step 3.

2.5a If ACCESS-TYPE not = 'RK1', then go to Step 2.5.

Generate DML for a primary or secondary key for which multiple values were specified:

2.5a.1 Set RK-INDEX = REC-SELECT-SPEC-PTR

2.5a.2 If RK-KEYCODE (RK-INDEX) = 'U' generate DML:

\[
\text{MVZ LOOP.i IF1 rk1-loop-max}
\]

2.5a.3 If RK-KEYCODE (RK-INDEX) = 'D' generate DML:

\[
\text{LOOP.i MVZ }
\]

Increment the loop labeler: \( i = i + 1 \)

\[
\text{LOOP.i IF1 rk1-loop-max}
\]

2.5a.4 Transform 'RK2' access specifications:

For each 'RK2' access specification generate DML:

\[
\text{IF2 rk2-loop-count MVK rk2-rk-index}
\]

2.5a.5 Transform the 'RK3' access specification:

2.5a.5.1 Find the record key components in the RECORD-KEY-TABLE:

For \( j = 1 \) to RK-DF-USED(RK-INDEX):
Set RS2-DFID(j) = RK-DFID(RK-INDEX, j)
RS2-VARIABLE = ISQL-VAR-n
where \( n = RK-ISQ-PTR \)

2.5a.5.2 If RK-KEYCODE(RK-INDEX) = 'U' generate DML:

17-3
FFR rs2-rtid,
  (rs2-dfid(j) =
   rs2-variable(j))
IFRECNOTFOUND
EXITLOOP.i
ENDIF

2.5a.5.3 If RK-KEYCODE(RK-INDEX) = 'D'
generate DML:

  FFR rs2-rtid,
    (rs2-dfid(j) =
     rs2-variable(j))
  rs2-lock
IFRECNOTFOUND
EXITLOOP.i
ENDIF

Push onto DML stack:

  ENDLOOP.i - 1
  ENDLOOP.i
  FNR rs2-rtid,
    (rs2-dfid(j) =
     rs2-variable(j))
  rs2-lock

Note that FNR is now on top of the stack.

2.5a.5.4 Go to Step 3.

2.5 If ACCESS-TYPE not = 'RA' or 'RAI', then generate an
error message and abandon the access path.

Generate DML for an area scan:

2.5.1 Increment the loop labeler: i=i+1

2.5.2 Generate DML:

  LOOP.i-1
  FFA ras-rtid, ras-areaid, ras-lock
  LOOP.i
  IFRECNOTFOUND
  EXITLOOP.i
  ENDIF

2.5.3 Push onto DML stack:

  17-4
2.5.4  If ACCESS-TYPE = 'RA'
Push onto DML stack:

FNA ras-rtid, ras-areaid, ras-lock

Note that the FNA is now on top of the stack if
ACCESS-TYPE = 'RA' and ENDLOOP.i is on top if
ACCESS-TYPE = 'RAI'.

3. Transform the rest of the access specifications for the
path, by doing the following for each of the
specifications in the path:

If ACCESS-TYPE = 'CAL', go to Step 3.18.
If ACCESS-TYPE = 'CIF', go to Step 3.21.
If ACCESS-TYPE = 'EI', go to Step 3.19.1.
If ACCESS-TYPE = 'EP', go to Step 3.19.2.
If ACCESS-TYPE = 'FG1', go to Step 3.14.1.
If ACCESS-TYPE = 'FG2', go to Step 3.14.2.
If ACCESS-TYPE = 'FG3', go to Step 3.14.3.
If ACCESS-TYPE = 'FG4', go to Step 3.14.4.
If ACCESS-TYPE = 'FU', go to Step 3.15.
If ACCESS-TYPE = 'FU1', go to Step 3.15.a.
If ACCESS-TYPE = 'FU2', go to Step 3.15.b.
If ACCESS-TYPE = 'FU3', go to Step 3.15.c.
If ACCESS-TYPE = 'FU4', go to Step 3.15.d.
If ACCESS-TYPE = 'IE', go to Step 3.17.
If ACCESS-TYPE = 'IF', go to Step 3.20.
If ACCESS-TYPE = 'IT1', go to Step 3.16.1.
If ACCESS-TYPE = 'IT2', go to Step 3.16.2.
If ACCESS-TYPE = 'MR1', go to Step 3.22.1.
If ACCESS-TYPE = 'MR2', go to Step 3.22.2.
If ACCESS-TYPE = 'MVS', go to Step 3.14.5.
If ACCESS-TYPE = 'NXS', go to Step 3.23.
If ACCESS-TYPE = 'OC1', go to Step 3.24.1.
If ACCESS-TYPE = 'OC2', go to Step 3.24.2.
If ACCESS-TYPE = 'OC3', go to Step 3.24.3.
If ACCESS-TYPE = 'OC4', go to Step 3.24.4.
If ACCESS-TYPE = 'OC5', go to Step 3.24.5.
If ACCESS-TYPE = 'OC6', go to Step 3.24.6.
If ACCESS-TYPE = 'OU4', go to Step 3.27.1.
If ACCESS-TYPE = 'OU5', go to Step 3.27.2.
If ACCESS-TYPE = 'PID', go to Step 3.25.
If ACCESS-TYPE = 'RC', go to Step 3.9.
If ACCESS-TYPE = 'RDK', go to Step 3.5.

17-5
If ACCESS-TYPE = 'RD2', go to Step 3.6.
If ACCESS-TYPE = 'RF1', go to Step 3.2.1.
If ACCESS-TYPE = 'RF2', go to Step 3.2.2.
If ACCESS-TYPE = 'RF3', go to Step 3.2.3.
If ACCESS-TYPE = 'RDK', go to Step 3.7.
If ACCESS-TYPE = 'RI2', go to Step 3.8.
If ACCESS-TYPE = 'RS1', go to Step 3.1.1.
If ACCESS-TYPE = 'RS3', go to Step 3.1.2.
If ACCESS-TYPE = 'RS4', go to Step 3.1.3.
If ACCESS-TYPE = 'RS5', go to Step 3.1.4.
If ACCESS-TYPE = 'RUK', go to Step 3.3.
If ACCESS-TYPE = 'RU2', go to Step 3.4.
If ACCESS-TYPE = 'SD', go to Step 3.12.
If ACCESS-TYPE = 'SI', go to Step 3.13.
If ACCESS-TYPE = 'SM1', go to Step 3.11.1.
If ACCESS-TYPE = 'SM2', go to Step 3.11.2.
If ACCESS-TYPE = 'SO1', go to Step 3.10.1.
If ACCESS-TYPE = 'SO2', go to Step 3.10.2.
If ACCESS-TYPE = 'UIF', go to Step 3.26.

If ACCESS-TYPE not = any of the above, generate an error message and abandon the access path.

3.1 Generate DML for record selection:

3.1.1 Generate DML for 'RS1'.

\[
\text{IF NOT} \; \text{rs1-rtidl, rs1-dfidl, rs1-op, rs1-rtidr, rs1-dfidr} \\
\text{NEXTINLOOP.i} \\
\text{ENDIF} \\
\]

Proceed with next iteration of Step 3.

3.1.2 Generate DML for 'RS3'.

\[
\text{IF NOT} \; \text{rs3-value, rs3-op, rs3-variable} \\
\text{NEXTINLOOP.i} \\
\text{ENDIF} \\
\]

Proceed with next iteration of Step 3.

3.1.3 Generate DML for 'RS4'.

\[
\text{IF NOT} \; \text{rs4-rtid, rs4-dfid, rs4-op, rs4-variable} \\
\text{NEXTINLOOP.i} \\
\text{ENDIF} \\
\]
Proceed with next iteration of Step 3.

3.1.4 Generate DML for RS5'.

IFC

Proceed with next iteration of Step 3.

3.2 Generate DML for function application:

3.2.1 Generate DML for 'RF1'.

GIF rfl-rtid, rfl-dfid, rfl-position

Proceed with next iteration of Step 3.

3.2.2 Generate DML for 'RF2'.

OU2 rf2-value, rf2-position

Proceed with next iteration of Step 3.

3.2.3 Generate DML for 'RF3'.

OU3 rf3-alg-id, rf3-mod-inst, rf3-parmno, rf3-is-ptr

Proceed with next iteration of Step 3.

3.3 Generate DML for 'RUK'.

Find the RK-REC-KEY entry in the RECORD-KEY-TABLE by setting RK-INDEX = REC-SELECT-SPEC-PTR.

Set RUK-KEYVALUE = the concatenation of ISQL-VAR-i through ISQ-VAR-j where i = RK-ISQ-PTR of RK-DATA-FIELD(RK-INDEX, L) and n = RK-ISQ-PTR of RK-DATA-FIELD(RK-INDEX, RK-DF-USED).

Generate DML for record update with direct access key:

RUK rk-rtid, ruk-keyvalue, ruk-lock

Proceed with next iteration of Step 3.

3.4 Generate DML for 'RU2'.

Generate DML for update to current of record type, which may be a member in set RU2-SETID:
RUS ru2-rtid, ru2-setid, ru2-lock

Proceed with next iteration of Step 3.

3.5 Generate DML for 'RDK'.

Find the RK-REC-KEY entry in the RECORD-KEY-TABLE by setting PK-INDEX = REC-SELECT-SPEC-PTR.

Set RDK-KEYVALUE = the concatenation of ISQL-VAR-i through ISQL-VAR-j where i = RK-ISQ-PTR of RK-DATA-FIELD(RK-INDEX,L) and n = RK-ISQ-PTR of RK-DATA-FIELD(RK-INDEX, RK-DF-USED).

Generate DML for record update with direct access key:

    RDK rk-rtid, rdk-keyvalue, rdk-lock

Proceed with next iteration of Step 3.

3.6 Generate DML for 'RD2'.

Generate DML for delete of current of record type, which may be a member in set RD2-SETID:

    RDS rd2-rtid, rd2-setid, rd2-lock

Proceed with next iteration of Step 3.

3.7 Generate DML for 'RIK'.

Find the RK-REC-KEY entry in the RECORD-KEY-TABLE by setting RK-INDEX = REC-SELECT-SPEC-PTR.

Set RDI-KEYVALUE = the concatenation of ISQL-VAR-i through ISQL-VAR-j where i = RK-ISQ-PTR of RK-DATA-FIELD(RK-INDEX,L) and n = RK-ISQ-PTR or RK-DATA-FIELD(RK-INDEX, RK-DF-USED).

Generate DML for record update with direct access key:

    RIK rk-rtid, rik-keyvalue, rik-lock

Proceed with next iteration of Step 3.

3.8 Generate DML for 'RI2'.

Generate DML for insert of record type, which may be

17-8
a member in set RI2-SETID:
RIS ri2-rtid, ri2-setid, ri2-lock
Proceed with next iteration of Step 3.

3.9 Generate DML for 'RC'.
Generate DML for record currency reset:
   RCR rcs-rtid
Proceed with next iteration of Step 3.

3.10 Generate DML for positioning on record set owner:
3.10.1 Generate DML for 'SO1'.
   Generate DML:
   FOW ss-setid
   IFRECFOUND
   Proceed with next iteration of Step 3.
3.10.2 Generate DML for 'SO2'.
   Generate DML:
   FOW ss-setid
   IFRECNOTFOUND
   NEXTINLOOP.i
   ENDIF
   Proceed with next iteration of Step 3.

3.11 Generate DML for positioning on record set member(s):
3.11.1 Generate DML for 'SM1'.
   FFM ss-setid, ss-rtid
   IFRECNOTFOUND
   Proceed with next iteration of Step 3.
3.11.2 Generate DML for 'SM2'.
   Increment loop labeler: i=i+1
Generate DML:

```
FFM ss-setid, ss-rtid
LOOP.i
IFRECNOTFOUND
EXITLOOP.i
ENDIF

Push onto DML stack:
ENDLOOP.i
FNM ss-setid, ss-rtid
```

Proceed with next iteration of Step 3.

3.12 Generate DML for 'SD'.

Generate DML for removal from record set:

```
SD sd-setid, sd-rtid
```

Proceed with next iteration of Step 3.

3.13 Generate DML for 'SI'.

Generate DML for insertion in record set:

```
SI si-setid, si-rtid
```

Proceed with next iteration of Step 3.

3.14 Generate DML for putting a value into a variable:

3.14.1 Generate DML for 'FG1'.

```
GIO fgl-rtid, fgl-dfid, fgl-variable
```

Proceed with next iteration of Step 3.

3.14.2 Generate DML for 'FG2'.

```
GF2 fg2-value, fg2-variable
```

Proceed with next iteration of Step 3.

3.14.3 Generate DML for 'FG3'.

```
MV3 fg3-alg-id, fg3-mod-inst, fg3-parm-no,
```

17-10
fg3-is-ptr
Proceed with next iteration of Step 3.

3.14.4 Generate DML for "FG4".

MV4 fg4-cma-constant, fg4-alg-id, fg4-mod-inst, fg4-parm-no
Proceed with next iteration of Step 3.

3.14.5 Generate DML for "MVS".

Generate DML for moving a run-time value to an ISQ variable:

MVS mvs-isq-ptr
Proceed with next iteration of Step 3.

3.15 Generate DML for "FU".

Generate DML for moving value of a variable to a field:

MV fus-variable, fus-rtid, fus-dfid
Proceed with next iteration of Step 3.

3.15a Generate DML for "FU1".

Generate DML for moving the value of a complex mapping algorithm parameter to a field:

MV6 ful-alg-id, ful-mod-inst, ful-parm-no, ful-dfno, ful-rtno
Proceed with next iteration of Step 3.

3.15b Generate DML for "FU2".

Generate DML for moving the values of a complex mapping algorithm parameter to an entire record:

MV5 fu2-alg-id, fu2-mod-inst, fu2-parm-no, fu2-rtno
Proceed with next iteration of Step 3.

3.15c Generate DML for "FU3".

17-11
Generate DML for moving value of a field to a complex mapping algorithm parameter:

MV7 fu3-dfno, fu3-rtno, fu3-alg-id, fu3-mod-inst, fu3-parm-no

Proceed with next iteration of Step 3.

3.15d Generate DML for 'FU4'.

Generate DML for moving an entire record to a complex mapping algorithm parameter:

MV8 fu4-rtno, fu4-alg-id, fu4-mod-inst, fu4-parm-no

Proceed with next iteration of Step 3.

3.16 Generate DML for IF condition THEN:

3.16.1 Generate DML for 'IT1'.

IF it1-rtidl, it1-dfidl, it1-op, it1-rtidr, it1-dfidr

Proceed with next iteration of Step 3.

3.16.2 Generate DML for 'IT2'.

IF it2-value, it2-op, it2-rtidr, it2-dfidr

Proceed with next iteration of Step 3.

3.17 Generate DML for 'IE'.

Generate DML for end of true part of IF statement:

ELSE

Proceed with next iteration of Step 3.

3.18 Generate DML for 'CAL'.

Generate DML to call a complex mapping algorithm for C-I or I-C conversions.

CALL cal-alg-id, cal-mod-inst, i, cal-parm-no (i) for i = 1 through cal-parm-count.

17-12
Proceed with next iteration of Step 3.

3.19 Generate DML for end of IF statement or of path:

3.19.1 Generate DML for 'EI'.

ENDIF

Proceed with next iteration of Step 3.

3.19.2 Generate DML for 'EP'.

Generate DML to empty buffer:

EP.i + 1

Pop the DML stack onto the generated DML access path

3.20 Generate DML to generate a COBOL IF statement in internal schema format for all ISQ-entries where ISQ-TYPE = 2 and ISQ-TYPE2-SOURCE = 'E' OR 'I'.

IIF
NXS
ELS
NLP i
EIF

Proceed with the next iteration of Step 3.

3.21 Generate DML to generate a COBOL IF statement in conceptual schema format for the WHERE clause.

CIF
NXS
ELS
NLP i
EIF

Proceed with the next iteration of Step 3.

3.22 Generate DML to move records from the schema area to working-storage area and back.

3.22.1 Generate DML for 'MR1'.

MR1 mr1-rtid, mr1-rtno
Proceed with next iteration of Step 3.

3.22.2 Generate DML for 'MR2'.

MR2 mr2-rtno, mr2-rtid

Proceed with next iteration of Step 3.

3.23 Generate DML to complete IF statement containing ORed conditions.

Generate DML for 'NXS'

NXS
ELS
NLP
ETF

3.24 Generate DML to retrieve from repeating data fields.

3.24.1 Generate DML for 'OC1' to initialize index to one:

SX1 oc1-dfno

Proceed with next iteration of Step 3.

3.24.2 Generate DML for 'OC2' to set index and index-max to user-specified value:

SX1 oc2-index-dfno, oc2-isq-ptr
MVI oc2-isq-ptr, oc2-dfno

Proceed with next iteration of Step 3.

3.24.3 Generate DML for 'OC3' to establish index-max value:

MVM oc3-index-dfno, oc3-occurs-dep-dfno, oc3-max-occurs

Proceed with next iteration of Step 3.

3.24.4 Generate DML for 'OC4' to establish loop construct for repeating fields:

LOP
IFX oc4-index-dfno
XLP.i

Push onto DML stack:

ELP.i
IX1 oc4-index-dfno

Note that IX1 is now on top of the stack. Proceed with next iteration of Step 3.

3.24.5 Generate DML for 'OC5' to output repeating field:

MVX oc5-dfno, oc5-num-indexes, oc5-idx-dfn01, oc5-idx-dfn02, oc5-idx-dfn03, oc5-is-ptr

Proceed with next iteration of Step 3.

3.24.6 Generate DML for 'OC6' to move null-values to repeating data fields:

MVY oc6-dfno, oc6-num-indexes oc6-idx-dfn0, oc6-idx-dfn2, oc6-idx-dfn3, oc6-type

3.25 Generate DML for 'PIO' to flush buffer.

PIO

3.26 Generate DML to generate a COBOL IF statement for ISQ-entries where ISQ-type = 2 and ISQ-TYPE2-SOURCE = 'U'.

UIF
NXS
ELS
NLP i
EIF

Proceed with next iteration of Step 3.

3.27 Generate DML to output where clause entries in CS format for evaluation at the conceptual schema level:

3.27.1 Generate DML for 'OU4' to move I-C output algorithm parameters to tags:

OU4 ou4-alg-id, ou4-mod-inst.
ou4-parm-no, ou4-tag-no

Proceed with the next iteration of Step 3.

3.27.2 Generate DML for 'OU5' to move data fields to tags:

OU5  ou5-dfno, ou5-datatype, ou5-tag-no

Proceed with the next iteration of Step 3.

4. When the entire access path has been transformed, return to PRE13 to have the appropriate version of PRE9 invoked.

Constraints

1. Not all types of access paths can be handled by this algorithm. The port access specification must be either an RK (direct to record given key value), an RK1 (direct to record given multiple values for key), or an RAS (area scan). These are the only types of port access specifications that should be produced by PRE6.

2. If the port access specification is an RK or RK1, then the selection criterion must be of the form:

   (rk-dfid = rk-variable)

   where:

   a. The set of rk-dfid's must be either a complete primary key or a complete secondary key

   b. The rk-dfid's need not be in "correct" physical sequence, but all must be specified that comprise the record key, i.e. generic keys are not supported.

   c. Primary and secondary record keys must be direct-access keys.

1.3 Outputs

The outputs of PRE7 are generic DML statements. The specific types of commands generated include the following.

BEGIN requests DBMS to start logging

17-16
CALL alg-id, {parm-id}

COMMIT

ELSE
ENDIF

ENDLOOP.i
EXITLOOP.i

RCR rtid

FFA rtid, areaid

FFM setid, rtid

FFR rtid, keyvalue

FNA rtid, areaid

FNM setid, rtid

FNR rtid

FOW setid

GIO rtid, dfid, variable

IF arg1, op, arg2
IFC

IFX conditional: IF index > index-max

IF1 conditional: IF loop count > number of values for a key

IF2 conditional: IF loop count = ith iteration

IFNOT arg1, op, arg2
IFRECFFOUND

IFRECFNOTFOUND

LOOP.i
MR1
MR2
  moves record from working-storage to schema area
MVK
  moves key field values to key field of record
MVM
  moves number of occurrences of occurrence depending on value to index-max
MVX
  moves indexed field to output
MV2
  moves zero to loop count controlling number of iterations through construct for multiple values of a key
MV variable, rtid, dfid
  moves value of variable into field dfid in record of type rtid
MV1
  moves value into variable
MV2 work-area, work-variable, variable
  moves value of work-variable in work-area into variable
MV3
  moves run-time value to complex mapping parameter for C-I conversion
MV4
  moves constant value to complex mapping parameter for C-I or I-C conversion
MV5
  moves complex mapping parameter to record after C-I conversion
MV6
  moves complex mapping parameter to data field of record after C-I conversion
MV7
  moves data field of record to complex mapping parameter for I-C conversion
MV8
  moves record to complex mapping parameter for I-C conversion
NEXTINLOOP.i
  transfers control to next iteration of i-th loop
OUTPUT arg, position
  moves arg to position in output buffer
PIO
  flushes output buffer
RDK rtid, keyvalue
  removes record of type rtid from database, using direct access by key value
RDS rtid, setid
  removes record of type rtid from database
RIK rtid, keyvalue
  adds record of type rtid to data-base, using direct access by key value
RIS rtid, setid
  adds record of type rtid to
ROLLBACK
data-base
requests DBMS to undo all changes
since transaction began
RUK rtid, keyvalue
modifies record of type rtid,
using direct access by key value
RUS rtid, setid
modifies record of type rtid
SD setid, rtid
removes record of type rtid from
its participation as a member in
record set setid
SI setid, rtid
inserts record of type rtid as a
member into record set setid

Specification of lock types (shared, exclusive, and none) are
carried explicitly on FFA, FNA, FFR, FNR, and some GIO commands.
Exclusive locks are carried explicitly on RDK, RDS, RIK, RIS,
RUK and RUS commands.

17.4 Internal Data Requirements
A stack of generated generic DML commands.
SECTION 13

FUNCTION PRE3 Generate CS/ES Transform

This function generates COBOL source code according to the ANSI X3.23-1974 standard which at runtime will transform from an aggregated, but not necessarily reduced, conceptual SELECT response to a completely reduced external response.

18.1 Inputs

1. **TARGET-HOST**   PIC XXX
   Host upon which the CS-ES Transform Program will execute at runtime.

2. **MY-HOST**   PIC XXX
   Host upon which CDPRE3 executes at precompile time.

3. **MOD-NAME**   PIC X(10)
   The program identification name of the CS-ES Transform Program.

4. **ES-ACTION-LIST** included in ESAL copy member
   External representation of fields to be retrieved.

5. **CS-ACTION-LIST** included in CSAL copy member
   Conceptual representation of fields to be retrieved.

6. **BOOLEAN-LIST** included in BOOLST copy member
   Contains information about boolean operators and parenthesized logic from the "WHERE" clause.

7. **CS-QUALIFY-LIST** included in CSQUAL copy member
   Conceptual representation of the "WHERE" clause.

8. **IS-QUALIFY-LIST**
   Internal representation of the WHERE clause.

9. **ERRFILE**   PIC X(30)
The file name to which user error messages are written.

10. CMA-FLAG    PIC 9

    If zero, don't use complex mapping algorithm logic.
    If non-zero, use complex mapping algorithm logic.

18.2 CDM Requirements

None

18.3 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 to be generated. This file must be closed prior to the CDMACR call. Upon return to CDPRE8, 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 CDPRE8 input parameter TARGET-HOST.

    MACRO-NAME contains the name of the macro to be generated, for example CTOE1.

    SUBSTITUTION-LIST is described by the following structure:

    01 SUBSTITUTION-LIST
<table>
<thead>
<tr>
<th>03</th>
<th>SL-USED</th>
<th>PIC 99.</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>SL-MAX</td>
<td>PIC 99.</td>
</tr>
<tr>
<td>03</td>
<td>SL-ROW-SIZE</td>
<td>PIC 99.</td>
</tr>
<tr>
<td>03</td>
<td>SL-ENTRY OCCURS 8 TIMES INDEXED BY SL-INDEX.</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>SL-PARAMETER</td>
<td>PIC X(30).</td>
</tr>
<tr>
<td>05</td>
<td>SL-SUBST-VAL</td>
<td>PIC X(30).</td>
</tr>
</tbody>
</table>

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, for example Pl. SL-SUBST-VAL (index) contains the corresponding substitution value, for example CS-NDML-NO.

18.4 Processing

1. Generate two unique file names to contain the generated COBOL code by calling GENFIL two times. GENFIL requires MY-HOST as an input parameter and returns the 30 character file name and the 5 character status.

   File 1 will contain code starting at the Identification Division. File 2 will contain code starting at the Linkage Section. These files will be appended at the end of CDPRE8 with the complete generated program residing in file 1 whose name will be placed in CDPRE8 output parameter GEN-FILE-NAME.

2. Generate IDENTIFICATION DIVISION by substituting the module name from input parameter MOD-NAME for Pl in macro CTOE1 on file 1.

3. Calculate the external schema record size by summing all used ES-SIZEs together. For each external field, add 1 additional position for the null flags. This is used to substitute for value P1 in macro CTOE2.

4. Determine which case is being handled. The case definitions are:

   CASE 1 - Requires two sorts in the CS-ES Transform Program

   Select DISTINCT with order by clause in which all sort fields are not projected.

   ex: Select DISTINCT :ul = coll
if the ES-DISTINCT-FLAG contains Y and if at least 1 field with ES-SORT-SEQUENCE greater than zero does not have a Y in its ES-PROJECT-FLAG, CASE 1 applies.

CASE 2 - Requires, at most, one sort in the CS-ES Transform Program.

Select with or without DISTINCT. Sort fields, if any, are all projected. Some fields, however, may not be projected if they are used only for qualification.

ex: Select :u1 = coll from Table

If all fields which have ES-SORT-SEQUENCE greater than zero also have ES-PROJECT-FLAG equal to Y, CASE 2 applies.

CASE 3 - No Sorts

Any Select with statistics functions.

ex: Select AVG (COL1) from Table

CASE 3 applies when any ES-FCTN-NAME is not blank.

5. Processing for CASE 1

5.1 Compute the conceptual schema record size by summing all used CS-SIZES together. For each conceptual field, add 1 additional position for the null flags.

5.2 Generate working storage records for ES-TEMP-REC, ES-RECORD-LENGTH and CS-REC and substitute for P1 the value computed in Step 3 in Macro CTOE2 on file 1.

5.3 For each CS field, generate on file 1 the CS null flags according to the following format:

05 CS-NULL-FLAG-xx PIC 9
5.4 Generate on file 1 each conceptual field description using the CS-TYPE, CS-SIZE and CS-ND fields. Use routine CDPIC to generate the picture clauses.

```
03 CS-VARxx pic clause.
...  
03 CS-VARyy pic clause.
```

where xx through yy are the CS-INDEX values and pic clause is the picture clause generated by CDPIC.

5.5 Generate a file to contain the sorted external schema output.

```
01 TEMP-REC PIC X(nn)
```

where nn is the sum computed in Step 3.

5.6 For each ES-ACTION entry, generate on file 1 a working storage external schema null flag and a working storage external schema field definition according to the following format.

```
01 WS-ES-REC.
  03 WS-NUL-FLAGS.
    05 WS-NUL-FLAG-01 PIC 9.
    ...  
    05 WS-NUL-FLAG-nn PIC 9.
    ...  
  03 WS-VAR-SS-01 pic clause.
    ...  
  03 WS-VAR-SS-nn pic clause.
```

where 01 to nn are the ES-INDEXEs and SS is the CS-NDML-NO. Use CDPIC to generate the variable picture
clauses using ES-SIZE, ES-TYPE and ES-ND.

5.7 For each ES-ACTION-LIST entry which has ES-PROJECT flag equal Y, generate working storage variables and null flag fields for use in duplicate elimination according to the following format:

```
01 OLDVAR-nn-mm-NUL  PIC 9.
01 OLDVAR-nn-mm pic clause.
```

where nn is the CS-NDML-NO and mm is the ES-INDEX value. Use CDPIC to generate the variable picture clauses using ES-SIZE, ES-TYPE and ES-ND.

5.8 Generate on file 1 the input parameters TARGET-HOST, by substituting the value of P1, MOD-NAME, by substituting the value of P2 and the length of the read buffer, by substituting value P3 into the CTOE4 macro.

5.9 Generate in file 1, the first part of the sort buffer for the distinct elimination sort using macro CTOE4B. For parameter P1, substitute the value 1. For parameter P2, substitute the value equal to twice the number of ES-PROJECT-FLAGS equal to Y.

5.10 Generate in file 1, the sort buffer elements for the distinct elimination sort. Scan the ES-ACTION-LIST. For each ES-PROJECT-FLAG equal to Y, generate two sort buffer elements using macro CTOE4C, one for the field's null flag and one for the ES field itself.

To generate the sort buffer for a projected field's null flag, use macro CTOE4C, substituting the value of ES-INDEX for P1 (sort key starting position), the value 1 for P2 (sort key length), the value N for P3 (sort key type) and the value A for P4 (ascending sort).

To generate the sort buffer for the projected field itself, a running total must be kept of ES-SIZES whether or not the field is projected. This value will be used in the calculation of the field's starting position. In macro CTOE4C, add 1, ES-USED and the running total described above to generate the value to substitute for P1.

As an example, suppose that there are two ES fields, the first one not projected and the second one

18-6
Since ES-PROJECT-FLAG (1) does not equal Y, add ES-SIZE (1) to the zero-initialized counter, giving the value 30. ES-PROJECT-FLAG (2) contains Y, so to generate the value for P1 in macro CTDE4C, add 1, ES-USED which has the value 2 and the counter value which is 30 giving 33. Therefore, 33 is the starting position of the second ES field and should be substituted for the parameter P1 in macro CTDE4C. Substitute the value of the current ES-SIZE for P2 (sort key length), the value of the current ES-TYPE for P3 (sort key type) and the value A for P4 (ascending sort).

5.11 Generate the first part of the sort buffer for the "order by" sort using macro CTDE4B into file 1. For parameter P1, substitute the value 2. For parameter P2, substitute the value equal to twice the number of ES-SORT-SEQUENCE values greater than zero. FILE-REC-KEY-USED is P2.

5.12 Generate the sort buffer elements for the "order by" sort into file 1. These elements must be generated for each ES field whose ES-SORT-SEQUENCE is greater than zero in ES-SORT-SEQUENCE order; that is the sort buffer elements associated with ES-SORT-SEQUENCE equal 1 must be generated before the sort buffer elements associated with ES-SORT-SEQUENCE equal 2. This is not necessarily the order in which the fields are encountered in the ES-ACTION-LIST. Two sort buffer elements must be generated for each ES field with ES-SORT-SEQUENCE greater than zero, a sort buffer element for the field's null flag and a sort buffer element for the ES field itself.

To generate the sort buffer element for the field's null flag (assuming the lowest numbered ES-SORT-SEQUENCE field greater than zero but not yet generated has been located), use macro CTDE4C, substituting the value of ES-INDEX for P1 (sort key starting position), the value 1 for P2 (sort key length), the value N for P3 (sort key type) and the value A for P4 (sort direction).
To generate the sort buffer for the sort field itself, a counter must be maintained which contains the sum of ES-SIZEs for those fields with a lesser ES-INDEX than the current field. Since generating these buffers will probably require multiple passes of the ES-ACTION-LIST, it may be advantageous to compute this sum after the sort field of interest has been located. As an example, suppose that the following ES-ACTION-LIST is encountered:

<table>
<thead>
<tr>
<th>ES-SORT-SEQUENCE (1)</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES-SORT-SEQUENCE (2)</td>
<td>0</td>
</tr>
<tr>
<td>ES-SORT-SEQUENCE (3)</td>
<td>1</td>
</tr>
<tr>
<td>ES-SIZE (1)</td>
<td>6</td>
</tr>
<tr>
<td>ES-SIZE (2)</td>
<td>30</td>
</tr>
<tr>
<td>ES-SIZE (3)</td>
<td>1</td>
</tr>
</tbody>
</table>

The sort buffer element associated with the field whose ES-INDEX is 3 must be generated first, because it contains the lowest ES-SORT-SEQUENCE greater than zero of any sort field not yet generated. Assuming that the field's null flag sort buffer element has been generated, the starting position of field 3 is the sum of the ES-SIZEs from fields 1 and 2 (36) plus the value of ES-USED which is 3 to account for the null flags plus 1 which equals 40. This is the value which must be substituted for P1 in macro CTOE4C for this field. Substitute the value of the current ES-SIZE for P2 (sort key length), the value of the current ES-TYPE for P3 (sort key type) and if the current ES-SORT-DIRECTION equals "A" or blank substitute "A" for P4 (sort direction), otherwise substitute "D" for P4.

5.13 Generate in file 2, the common linkage section using macro CTOE5. This macro has no parameters.

5.14 Generate in file 2, the linkage section ES variable descriptions for the projected fields. Use routine CDP8A, sending it the ES-ACTION-LIST, the ES-ACTION-LIST and the name of the closed file 2 as parameters. CDP8A will generate ES variable names and pictures according to the following format:

```
03 ES-VAR-esndml-esindexaa pic clause.
```

18-9
5.15 Generate on file 2 the names and picture clauses for the conceptual schema qualify variables which will be passed to the generated program at runtime.

In all cases, generate the following line:

```
01 CS-QUALIFY-VAR.
```

Scan the CS-QUALIFY-LIST searching for a zero value in a used CSQ-AUCR. If none are found, generate the following:

```
03 FILLER PIC X
```

For each used CSQ element with CSQ-AUCR equal zero, generate the following:

```
03 CSQ-VAR-nn picture clause.
```

where nn is the CSQ-INDEX of the current field.

Call CDPIC using the corresponding CSQ-L-TYPE, CSQ-L-SIZE and CSQ-L-ND to generate the picture clause.

5.16 Generate on file 2 the beginning of the procedure division using macro CTOE6C. This macro has no parameters.

5.17 If any used ISQ-EVAL-FLAG has a zero value, call CDGENIF to generate on file 2 the IF clauses to perform the final qualification on the returned conceptual rows. CDGENIF requires the following parameters:

Inputs
- BOOLEAN-LIST
- CS-QUALIFY-LIST
- DUMMY PIC X
- QUALIFY-TYPE PIC X VALUE "C"
- FILE-NAME PIC X(30)
- SUBTRANS-ID PIC 999 VALUE ZERO
- DUMMY PIC X

Outputs
- RET-STATUS PIC X(5)
FILE-NAME must contain the file name of the closed file 2.

If CDGENIF is successful (RET-STATUS equals KES-SUCCESSFUL), generate on the reopened file 2, the macro "CTOE18" which has no parameters. This macro terminates the IF clauses generated by CDGENIF.

5.18 Call CDCE to generate in file 2 calls to user modules to perform complex CS-ES transformations, if any are defined. Also, if user CS-ES transformation modules are defined, CDCE will generate into file 1 the names and descriptions of the parameters to be sent to the user module at runtime.

For those CS fields which have no complex CS-ES algorithm defined, CDCE will, for CASE 1 CS-ES programs, generate into file 2 moves from the CS variable names to working storage variable names previously generated. The null flag values are passed along as well.

The calling sequence for CDCE is:

**Inputs**

<table>
<thead>
<tr>
<th></th>
<th>WORK-FILE1</th>
<th>PIC X(30)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WORK-FILE2</td>
<td>PIC X(30)</td>
</tr>
<tr>
<td></td>
<td>STRAIGHT-MOVE-FLAG</td>
<td>PIC X VALUE &quot;N&quot;</td>
</tr>
<tr>
<td></td>
<td>CS-ACTION-LIST</td>
<td>COPY CSAL OF IISSCLIB</td>
</tr>
<tr>
<td></td>
<td>ES-ACTION-LIST</td>
<td>COPY ESAL OF IISSCLIB</td>
</tr>
<tr>
<td></td>
<td>TARGET-HOST</td>
<td>PIC XXX</td>
</tr>
<tr>
<td></td>
<td>CMA-FLAG</td>
<td>PIC 9.</td>
</tr>
</tbody>
</table>

**Outputs**

<table>
<thead>
<tr>
<th></th>
<th>RET-STATUS</th>
<th>PIC X(5)</th>
</tr>
</thead>
</table>

WORK-FILE1 must contain the name of the closed file 1. WORK-FILE2 must contain the name of the closed file 2. TARGET-HOST is the CDPRE8 input parameter.

5.19 Generate on file 2 the write of the temporary ES file using macro CTOE5A. This macro has no parameters.

5.20 Generate on file 2 the call to the sort routine for the duplicate elimination sort and the reading of the results file using macro CTOE6C1. This macro has no parameters.
5.21 Generate on file 2 the projected field duplicate elimination by:

5.21.1 Generating the following 1 line:

\[
\text{IF FIRST-RECORD NOT} = 1
\]

5.21.2 Generating the comparison of null flags by scanning the ES-ACTION-LIST. For each ES field which has ES-PROJECT-FLAG equal Y, generate 1 line as follows:

\[
\text{AND OLDVAR-ndml-esindex-NULL} = \text{WS-NULL-FLAG-esindex}
\]

where ndml is the current value of CS-NDML-NO and esindex is the current ES-INDEX value.

5.21.3 After all null flag comparisons are generated, scan the ES-ACTION-LIST again. For each ES field that has ES-PROJECT-FLAG equal Y, generate 1 line as follows:

\[
\text{AND OLDVAR-ndml-esindex} = \text{WS-VAR-ndml-esindex}
\]

where ndml is the current value of CS-NDML-NO and esindex is the current ES-INDEX value.

5.21.4 After all of the projected flag and field comparisons have been generated, generate the following line:

\[
\text{GO TO RELEASE-RECORDS.}
\]

5.22 Generate into file 2 the moves from the working storage fields and flags to the oldvar fields and flags for the next iteration of the distinct test.

Scan the ES-ACTION-LIST. For each ES field that has ES-PROJECT-FLAG equal Y, generate 2 move statements as follows:

\[
\begin{align*}
\text{MOVE WS-VAR-ndml-esindex} & \text{ TO OLDVAR-ndml-esindex;} \\
\text{MOVE WS-NULL-FLAG-esindex} & \text{ TO OLDVAR-ndml-esindex-NULL.}
\end{align*}
\]

where ndml is the value of CS-NDML-NO and esindex is the value of the current ES-INDEX.
5.23 Generate on file 2 the end of the release records loop using macro CTOE19. This macro has no parameters.

5.24 Generate into file 2 the call to the "order by" sort and the read loop using macro CTOE6D. This macro has no parameters.

5.25 Generate into file 2 the projection step which places projected fields and flags into the output file.

Scan the ES-ACTION-LIST. For each ES field which has ES-PROJECT-FLAG equal Y, generate 2 move statements as follows:

\[
\begin{align*}
\text{MOVE} & \quad \text{WS-VAR-ndml-esindex TO ES-VAR-ndml-esindex} \\
\text{MOVE} & \quad \text{WS-NULL-FLAG-esindex TO ES-NULL-FLAG-esindex}
\end{align*}
\]

where esindex is the value of the current ES-INDEX

5.26 Generate into file 2 the EXIT-PROGRAM and part of the DEL-PARA paragraphs using macro CTOE14, substituting a blank character for parameter P1.

5.27 Generate on file 2 two calls to "DELFIL" to delete ES-TEMP and TEMP-FILE as follows:

\[
\begin{align*}
\text{CALL} & \quad \text{"DELFIL" USING MY-HOST, CDMESRES.} \\
\text{CALL} & \quad \text{"DELFIL" USING MY-HOST, CDMTMPFL.}
\end{align*}
\]

5.28 Append file 2 to file 1 by calling CDCWF after closing both files. CDCWF requires the following parameters:

\[
\begin{align*}
\text{FILE 1} & \quad \text{PIC X(30)} \\
\text{FILE 2} & \quad \text{PIC X(30)} \\
\text{MY-HOST} & \quad \text{PIC XXX}
\end{align*}
\]

Upon return from CDCWF, file 1 will contain the complete generated program and file 2 will not exist (CDCWF deletes it). Move the name of file 1 to the CDPRE8 output parameter GEN-FILE-NAME. Case 1 processing is now complete.

6. Processing for CASE 2

6.1 Compute the conceptual schema record size by summing all used CS-SIZEs together. For each conceptual field, add 1 additional position for the null flags.
6.2 Generate working storage records for ES-TEMP-REC, ES-RECORD-LENGTH and CS-REC and substitute for P1 the value computed in Step 3 in macro CTOE2 on file 1.

6.3 If any used ES-SORT-SEQUENCEs are greater than zero or if ES-DISTINCT-FLAG equals Y, generate on file 1 a temporary file to contain the sorted External Schema output.

   01 TEMP-REC PIC(nn)

where

   nn is the sum computed in Step 3.

6.4 For each CS field, generate on file 1 the CS null flags according to the following format:

   05 CS-NUL1-FLAG-xx PIC 9.
   .
   .
   05 CS-NUL1-FLAG-yy PIC 9.

where xx through yy are the values of CS-INDEX. The 05 must start in column 16.

6.5 Generate on file 1 each conceptual field description using the CS-TYPE, CS-SIZE and CS-ND fields. Use routine CDPIC to generate the picture clauses.

   03 CS-VARxx pic clause.
   .
   .
   03 CS-VARyy pic clause.

where xx through yy are the values of CS-INDEX and pic clause is the picture clause generated by CDPIC.

6.6 Generate on file 1 the common working storage section by substituting the value of input parameter TARGET-HOST for P1, the value of the input parameter MOD-NAME for P2 into the CTOE4 macro and the value for the length of the read buffer for P3.

6.7 If any used ES-SORT-SEQUENCE numbers are greater than zero or if ES-DISTINCT-FLAG equals Y, generate on file
1, for each ES-ACTION entry, a working storage external schema null flag and a working storage external schema field definition according to the following format.

```
01 WS-ES-REC.
  03 WS-NUL-FLAG.
  05 WS-NUL-FLAG-01 PIC 9.
      .
  05 WS-NUL-FLAG-nn PIC 9.
  03 WS-VAR-SS-01 pic clause.
      .
  03 WS-VAR-SS-nn pic clause.
```

where 01 to nn are the ES-INDEXes and SS is the CS-NDML-NO. Use CDPIC to generate the variable picture clauses using ES-SIZE, ES-TYPE and ES-ND.

6.8 If ES-DISTINCT-FLAG equals Y, generate in file 1 the following one line which will serve as a comparison buffer for duplicate elimination.

```
01 DST-REC PIC X(nnn)
```

where nnn is the value computed in Step 3.

6.9 If any used ES-SORT-SEQUENCE is greater than zero or ES-DISTINCT equals Y, a sort buffer must be generated on file 1. To generate the first part of the sort buffer, use macro CTOE4B substituting the value 1 for P1 and two times the value of ES-USED for P2.

6.10 If any used ES-SORT-SEQUENCE is greater than zero, generate in file 1 the sort buffer elements for the "order by" portion of the sort.

These elements must be generated for each ES field whose ES-SORT-SEQUENCE is greater than zero in ES-SORT-SEQUENCE order; that is the sort buffer elements associated with ES-SORT-SEQUENCE equal 1 must be generated before the sort buffer elements associated with ES-SORT-SEQUENCE equal 2. This is not necessarily the order in which the fields are encountered in the ES-ACTION-LIST. Two sort buffer elements must be generated for each ES field with
ES-SORT-SEQUENCE greater than zero, a sort buffer element for the field's null flag and a sort buffer element for the ES field itself.

To generate the sort buffer element for the field's null flag (assuming the lowest numbered ES-SORT-SEQUENCE field greater than zero, but not yet generated, has been located), use macro CTOE4C, substituting the value of ES-INDEX for P1 (sort key starting position), the value 1 for P2 (sort key length), the value N for P3 (sort key type) and the value A for P4 (sort direction).

To generate the sort buffer for the sort field itself, a counter must be maintained which contains the sum of ES-SIZES for those fields with a lesser ES-INDEX than the current field. Since generating these buffers will probably require multiple passes of the ES-ACTION-LIST, it may be advantageous to compute this sum after the sort field of interest has been located. As an example, suppose that the following ES-ACTION-LIST is encountered:

<table>
<thead>
<tr>
<th>ES-SORT-SEQUENCE (1)</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES-SORT-SEQUENCE (2)</td>
<td>0</td>
</tr>
<tr>
<td>ES-SORT-SEQUENCE (3)</td>
<td>1</td>
</tr>
<tr>
<td>ES-SIZE (1)</td>
<td>6</td>
</tr>
<tr>
<td>ES-SIZE (2)</td>
<td>30</td>
</tr>
<tr>
<td>ES-SIZE (3)</td>
<td>1</td>
</tr>
</tbody>
</table>

The sort buffer element associated with the field whose ES-INDEX is 3 must be generated first, because it contains the lowest ES-SORT-SEQUENCE greater than zero of any sort field not yet generated. Assuming that the field's null flag sort buffer element has been generated, the starting position of field 3 is the sum of the ES-SIZEs from fields 1 and 2 (36) plus the value of ES-USED which is 3 (to account for the null flags) plus 1 which equals 40. This is the value which must be substituted for P1 in macro CTOE4C for this field. Substitute the value of the current ES-SIZE for P2 (sort key length), the value of the current ES-TYPE for P3 (sort key type) and if the current ES-SORT-DIRECTION equals "A" or blank, substitute "A" for P4 (sort direction), otherwise substitute "D" for P4.

6.11 If ES-DISTINCT-FLAG equals Y, sort buffer elements must be generated on file 1 for any used ES fields.
which have ES-SORT-SEQUENCE equal zero. These sort buffer elements can be generated in the order of their occurrence on the ES-ACTION-LIST.

For each ES field which qualifies, generate two sort buffer elements using macro CTOE4C, one for the field's null flag and one for the ES field itself.

To generate the sort buffer for a field's null flag, use macro CTOE4C, substituting the value of ES-INDEX for P1 (sort key starting position), the value 1 for P2 (sort key length), the value N for P3 (sort key type) and the value A for P4 (ascending sort).

To generate the sort buffer for the field itself, a running total must be kept of ES-SIZEs. This value will be used in the calculation of the field's starting position. In macro CTOE4C, add 1, ES-USED and the running total described above to generate the value to substitute for P1.

As an example, suppose that the following ES-ACTION-LIST is encountered:

<table>
<thead>
<tr>
<th>ES-SORT-SEQUENCE</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>2</td>
</tr>
<tr>
<td>(2)</td>
<td>1</td>
</tr>
<tr>
<td>(3)</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ES-SIZE</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>30</td>
</tr>
<tr>
<td>(2)</td>
<td>6</td>
</tr>
<tr>
<td>(3)</td>
<td>30</td>
</tr>
</tbody>
</table>

The sort buffer elements have already been generated for the first two ES fields in Step 5.11. However, while scanning the ES-ACTION-LIST searching for a used field with ES-SORT-SEQUENCE equal zero, add the ES-SIZEs to a zero initialized counter. Add 30 for field 1 and 6 for field 2 giving 36. When a field is encountered with ES-SORT-SEQUENCE equal zero, add the counter contents (36) plus ES-USED (3) plus 1 giving 40, the value to substitute for P1 in macro CTOE4C.

Substitute the value of the current ES-SIZE for P2 (sort key length), the value of the current ES-TYPE for P3 (sort key type) and the value A for P4 (ascending sort).

6.12 Generate in file 2 the common linkage section using macro CTOE5. This macro has no parameters.
6.13 Generate in file 2 the linkage section ES variable descriptions for the projected fields. Use routine CDP8A, sending it the CS-ACTION-LIST, the ES-ACTION-LIST and the name of the closed file 2 as parameters. CDP8A will generate ES variable names and pictures according to the following format:

\[
03 \text{ES-VAR-csndml-esindexaa picture clause.}
\]

\[
\ldots
\]

\[
03 \text{ES-VAR-csndml-esindexnn picture clause.}
\]

6.14 Generate on file 2 the names and picture clauses for the conceptual schema qualify variables which will be passed to the generated program at runtime.

In all cases, generate the following line:

\[
01 \text{CS-QUALIFY-VAR.}
\]

Scan the CS-QUALIFY-LIST searching for a zero value in a used CSQ-AUCR. If none are found, generate the following:

\[
03 \text{FILLER PIC X.}
\]

For each used CSQ element with CSQ-AUCR equal zero, generate the following:

\[
03 \text{CSQ-VAR-nn picture clause.}
\]

where nn is the CSQ-INDEX of the current field.

Call CDPIC using the corresponding CSQ-L-TYPE, CSQ-L-SIZE and CSQ-L-ND to generate the picture clause.

6.15 If any used ES-SORT-SEQUENCE is greater than zero or ES-DISTINCT-FLAG equals Y, generate on file 2 the beginning of the Procedure Division using macro CTOE6B. This macro has no parameters.

6.16 If neither of the conditions in the previous step hold, generate on file 2 the beginning of the Procedure Division using macro CTOE6. This macro has no parameters.

6.17 If any used ISQ-EVAL-FLAG has a zero value, call
CDGENIF to generate on file 2 the IF clauses to perform the final qualification on the returned conceptual rows. CDGENIF requires the following parameters:

**Inputs**
- BOOLEAN-LIST
- CS-QUALIFY-LIST
- DUMMY PIC X
- QUALIFY-TYPE PIC X VALUE "C"
- FILE-NAME PIC X(30)
- SUBTRANS-ID PIC 999 VALUE ZERO
- DUMMY PIC X

**Outputs**
- RET-STATUS PIC X(5)

FILE-NAME must contain the file name of the closed file 2.

IF CDGENIF is successful (RET-STATUS equals KES-SUCCESSFUL), generate on the reopened file 2 the macro CTOE18 which has no parameters. This macro terminates the IF clauses generated by CDGENIF.

6.18 Call CDCE to generate in file 2 calls to user modules to perform complex and non-complex CS-ES transformations, if any are defined. Also, if complex CS-ES transformation modules are defined, CDCE will generate into file 1 the names and descriptions of the parameters to be sent to the user module at runtime.

The calling sequence for CDCE is:

**Inputs**
- WORK-FILE1 PIC X(30)
- WORK-FILE2 PIC X(30)
- STRAIGHT-MOVE-FLAG PIC X
- CS-ACTION-LIST COPY CSAL OF IISSCLIB
- ES-ACTION-LIST COPY ESAL OF IISSCLIB
- TARGET-HOST PIC XXX
- CMA-FLAG PIC 9.

**Outputs**
- RET-STATUS PIC X(5)

WORK-FILE1 must contain the name of the closed file 1.
WORK-FILE2 must contain the name of the closed file 2.
TARGET-HOST is the CDPRE3 input parameter.

The STRAIGHT-MOVE-FLAG must be set to Y if any used ES-SORT-SEQUENCE is greater than zero or if ES-DISTINCT-FLAG equals Y. If neither of the previous conditions hold, set STRAIGHT-MOVE-FLAG to N.

6.19 If any used ES-SORT-SEQUENCE is greater than zero or ES-DISTINCT-FLAG equals Y, generate on file 2 the write of the temporary file using the CTOE5A macro which has no parameters.

6.20 If any used ES-SORT-SEQUENCE is greater than zero or ES-DISTINCT-FLAG equals Y, generate on file 2 the sort call and read loop using the CTOE6B1 macro which has no parameters.

6.21 If ES-DISTINCT-FLAG equals Y, generate on file 2 the duplicate elimination Procedure Division code using the CTOE20 macro which has no parameters.

6.22 If any used ES-SORT-SEQUENCE is greater than zero or ES-DISTINCT-FLAG equals Y, generate into file 2, a projection step which places projected fields and flags into the output parameters.

Scan the ES-ACTION-LIST. For each ES field which has ES-PROJECT-FLAG equal Y, generate 2 move statements as follows:

```
MOVE WS-VAR-ndml-esindex TO ES-VAR-ndml-esindex
MOVE WS-NULL-FLAG-esindex TO ES-NULL-FLAG-esindex
```

where esindex is the value of the current ES-INDEX.

6.23 Generate into file 2 the EXIT-PROGRAM and part of the DEL-PARA paragraphs using macro CTOE14, substituting a blank character for parameter P1.

6.24 If any used ES-SORT-SEQUENCE is greater than zero or ES-DISTINCT-FLAG equals Y, generate on file 2 two calls to DELFIL to delete ES-TEMP and TEMP-FILE as follows:

```
CALL "DELFIL" USING MY-HOST, CDMESRES.
CALL "DELFIL" USING MY-HOST, CDMTMPFL.
```

6.25 Append file 2 to file 1 by calling CDCWF after closing both files. CDCWF requires the following parameters:
FILE1  PIC X(30)
FILE 2  PIC X(30)
MY-HOST  PIC XXX

Upon return from CDCWF, file 1 will contain the complete generated program and file 2 will not exist (CDCWF deletes it). Move the name of file 1 to the CDPRE8 output parameter GEN-FILE-NAME. Case 2 processing is now complete.

7. Processing for CASE 3

7.1 Compute the conceptual schema record size by summing all used CS-SIZEs together. For each conceptual field, add 1 additional position for the null flags.

7.2 Generate working storage records for ES-TEMP-REC, ES-RECORD-LENGTH, and CS-REC by substituting for P1 the value computed in step 3 in macro CTOE2 on file 1.

7.3 For each CS field, generate on file 1 the CS null flags according to the following format:

```
05  CS-NUL xx FLAG-xx  PIC 9.

05  CS-NUL yy FLAG-yy  PIC 9.
```

where xx through yy are the values of CS-INDEX. The 05 must start in column 16.

7.4 Generate on file 1 each conceptual field description using the CS-TYPE, CS-SIZE and CS-ND fields. Use routine CDPIC to generate the picture clauses.

```
03  CS-VARxx  PIC clause.

03  CS-VARyy  PIC clause.
```

where xx through yy are the values of CS-INDEX and pic clause is the picture clause generated by CDPIC.

7.5 If any used ES-FCTN-DISTINCT equals Y, the following temporary record is constructed:

18-20
01 TEMP-REC PIC X(nn).

where

nn is the sum computed in Step 3.

7.6 Generate on file 1 working storage records by substituting the value of input parameter TARGET-HOST for P1, the value of the input parameter MOD-NAME for P2 and the length of the read buffer for P3 into the CTOE4 macro.

7.7 If any used ES-FCTN-DISTINCT equals Y, scan the ES-ACTION-LIST searching for the largest ES-SIZE which has ES-FCTN-DISTINCT equal Y. Generate on file 1 the distinct elimination working storage elements by substituting for parameter P1, the maximum of the largest ES-SIZE with ES-FCTN-DISTINCT equal Y or 18 in macro CTOE4A.

7.8 If no used ES-FCTN-DISTINCT equals Y, scan the ES-ACTION-LIST searching for the largest used ES-SIZE. Generate on file 1 the non-distinct working storage elements by substituting, for parameter P1, the maximum of the largest ES-SIZE or 18 in macro CTOE3.

7.9 For each ES-ACTION entry, generate on file 1 a working storage external schema null flag and a working storage external schema field definition according to the following format.

01 WS-ES-REC.
   03 WS-NULL-FLAGS.
      05 WS-NULL-FLAG-01 PIC 9.
      .
      .
      05 WS-NULL-FLAG-nn PIC 9.
   03 WS-VAR-SS-01 pic clause.
      .
      .
   03 WS-VAR-SS-nn pic clause.

where 01 to nn are the ES-INDEXes and SS is the CS-NDML-NO. Use CDPIC to generate the variable 18-21
picture clauses using ES-SIZE, ES-TYPE and ES-ND.

7.10 Generate in file 2 the common linkage section using macro CTOE5. This macro has no parameters.

7.11 Generate in file 2 the linkage section ES variable descriptions for the output fields. Use routine CDP8A, sending it the CS-ACTION-LIST, the ES-ACTION-LIST and the name of the closed file 2 as parameters. CDP8A will generate ES variable names and pictures according to the following format:

\[
03 \text{ ES-VAR-csndml-esindexaa picture clause.} \\
\vdots \\
03 \text{ ES-VAR-csndml-esindexnn picture clause.}
\]

7.12 Generate on file 2 the names and picture clauses for the conceptual schema qualify variables which will be passed to the generated program at runtime.

In all cases, generate the following line:

\[
01 \text{ CS-QUALIFY-VAR.}
\]

Scan the CS-QUALIFY-LIST searching for a zero value in a used CSQ-AUCR. If none are found, generate the following:

\[
03 \text{ FILLER PIC X.}
\]

For each used CSQ element with CSQ-AUCR equal zero, generate the following:

\[
03 \text{ CSQ-VAR-nn picture clause.}
\]

where nn is the CSQ-INDEX of the current field. Call CDPIC using the corresponding CSQ-L-TYPE, CSQ-L-SIZE and CSQ-L-ND to generate the picture clause.

7.13 Generate on file 2 the beginning of the Procedure Division using macro CTOE6A which has no parameters.

7.14 If any used ISQ-EVAL-FLAG has a zero value, call CDGENIF to generate on file 2 the IF clauses to perform the final qualification on the returned conceptual rows. CDGENIF requires the following parameters:
Inputs

BOOLEAL-LIST
CS-QUALIFY-LIST
DUMMY PIC X
QUALIFY-TYPE PIC X VALUE "C"
FILE-NAME PIC X(30)
SUBTRANS-ID PIC 999 VALUE ZERO
DUMMY PIC X

Outputs

RET-STATUS PIC X(5)

FILE-NAME must contain the file name of the closed file 2.

If CDGENIF is successful (RET-STATUS equals KES-SUCCESSFUL), generate on the reopened file 2 the macro CTOE18 which has no parameters. This macro terminates the IF clauses generated by CDGENIF.

7.15 Call CDCE to generate in file 2 CS-ES transformations. Also, if complex CS-ES transformation modules are defined, CDCE will generate into file 1 the names and descriptions of the parameters to be sent to the user module at runtime.

The calling sequence for CDCE is:

Inputs

01 WORK-FILE1 PIC X(30)
01 WORK-FILE2 PIC X(30)
01 STRAIGHT-MOVE-FLAG PIC X VALUE "N"
01 CS-ACTION-LIST COPY CSAL OF IISSCLIB
01 ES-ACTION-LIST COPY ESAL OF IISSCLIB
01 TARGET-HOST PIC XXX
01 CMA-FLAG PIC 9

Outputs

01 RET-STATUS PIC X(5)

WORK-FILE1 must contain the name of the closed file 1.
WORK-FILE2 must contain the name of the closed file 2.
TARGET-HOST is the CDPRES input parameter.

7.16 Generate on file 2 the write of the temporary ES file using macro CTOE5A. This macro has no parameters.

7.17 For each ES-ACTION-LIST entry, generate in file 2 the
Procedure Division function logic as detailed for each function type below.

### 7.17.1 COUNT DISTINCT

If ES-FCTN-NAME equals COUNT and ES-FCTN-DISTINCT equals Y, substitute the following values into macro CTOE7.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Substitution Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>ES-INDEX value</td>
</tr>
<tr>
<td>P2</td>
<td>CS-NDML-NO value</td>
</tr>
</tbody>
</table>
| P3        | If ES-TYPE equals C, substitute X  
|           | If ES-TYPE does not equal C, substitute N  |

### 7.17.2 SUM DISTINCT

If ES-FCTN-NAME equals SUM and ES-FCTN-DISTINCT equals Y, substitute the following values into macro CTOE8.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Substitution Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>ES-INDEX value</td>
</tr>
<tr>
<td>P2</td>
<td>CS-NDML-NO value</td>
</tr>
<tr>
<td>P3</td>
<td>If ES-TYPE equals C, substitute X</td>
</tr>
</tbody>
</table>

### 7.17.3 AVG DISTINCT or MEAN DISTINCT

If ES-FCTN-NAME equals AVG or MEAN and ES-FCTN-DISTINCT equals Y, substitute the following values into macro CTOE9.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Substitution Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>ES-INDEX value</td>
</tr>
<tr>
<td>P2</td>
<td>CS-NDML-NO value</td>
</tr>
<tr>
<td>P3</td>
<td>If ES-TYPE equals C, substitute X</td>
</tr>
</tbody>
</table>

### 7.17.4 COUNT

18-24
If ES-FCTN-NAME equals COUNT and ES-FCTN-DISTINCT does not equal Y, substitute the following values into macro CTOE10.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Substitution Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>ES-INDEX value</td>
</tr>
<tr>
<td>P2</td>
<td>CS-NDML-NO value</td>
</tr>
</tbody>
</table>

7.17.5 SUM

If ES-FCTN-NAME equals SUM and ES-FCTN-DISTINCT does not equal Y, substitute the following values into macro CTOE11.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Substitution Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>ES-INDEX value</td>
</tr>
<tr>
<td>P2</td>
<td>CS-NDML-NO value</td>
</tr>
</tbody>
</table>

7.17.6 AVG or MEAN

If ES-FCTN-NAME equals AVG or MEAN and ES-FCTN-DISTINCT does not equal Y, substitute the following values into macro CTOE12.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Substitution Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>ES-INDEX value</td>
</tr>
<tr>
<td>P2</td>
<td>CS-NDML-NO value</td>
</tr>
</tbody>
</table>

7.17.7 If ES-FCTN-NAME equals MIN, substitute the following values into macro CTOE13.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Substitution Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>ES-INDEX value</td>
</tr>
<tr>
<td>P2</td>
<td>CS-NDML-NO value</td>
</tr>
<tr>
<td>P3</td>
<td>If ES-TYPE equals C, substitute X</td>
</tr>
<tr>
<td></td>
<td>If ES-TYPE does not equal C, substitute :</td>
</tr>
<tr>
<td>P4</td>
<td>If ES-TYPE equals S, substitute 999999999999999999 (18 nines)</td>
</tr>
<tr>
<td></td>
<td>If ES-TYPE does not equal 18-25</td>
</tr>
</tbody>
</table>

18-25
If ES-FCTN-NAME equals MAX, substitute the following values into macro CTOE13.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>ES-INDEX value</td>
</tr>
<tr>
<td>P2</td>
<td>CS-NDML-NO value</td>
</tr>
<tr>
<td>P3</td>
<td>If ES-TYPE equals C, substitute X. If ES-TYPE does not equal C, substitute N.</td>
</tr>
<tr>
<td>P4</td>
<td>If ES-TYPE equals S, substitute -99999999999999999 (minus followed by 17 nines). If ES-TYPE does not equal S, substitute the character string LOW-VALUE GREATER.</td>
</tr>
<tr>
<td>P5</td>
<td>If ES-TYPE equals S, substitute VARN. If ES-TYPE does not equal S, substitute VAR.</td>
</tr>
</tbody>
</table>

7.18 Generate in file 2 the EXIT-PROGRAM and part of the DEL-PARA paragraphs using macro CTOE14, substituting * for P1.

7.19 Generate in file 2 the following DELFIL call.

CALL "DELFIL" USING MY-HOST, CDMESRES.

7.20 If any used ES-FCTN-DISTINCT equals Y, generate in file 2 the following DELFIL call.

CALL "DELFIL" USING MY-HOST, CDMTMPFL.

7.21 If any used ES-FCTN-DISTINCT equals Y, generate the distinct elimination Procedure Division logic using macro CTOE15 which has no parameters.
7.22 Append file 2 to file 1 by calling CDCWF after closing both files. CDCWF requires the following parameters:

```
FILE1    PIC X(30)
FILE2    PIC X(30)
MY-HOST  PIC XXX
```

Upon return from CDCWF, file 1 will contain the complete generated program and file 2 will not exist (CDCWF deletes it). Move the name of file 1 to the CDPRE8 output parameter GEN-FILE-NAME. CASE 3 processing is now complete.

18.5 Outputs

1. GEN-FILE-NAME    PIC X(30)

   The file name containing the generated COBOL CS-ES transform program.

2. RET-STATUS    PIC X(5)

   Error Status - A value equal to KES-SUCCESSFUL as defined in copy member ERRCDM indicates successful completion.
LIBRARY NAME: VAX

MACRO NAME: CTOE1

PARAMETER: P1

IDENTIFICATION DIVISION.

PROGRAM-ID. P1.

* DESCRIPTION: THIS PROGRAM TRANSFORMS RETRIEVED CONCEPTUAL DATA TO EXTERNAL FORMAT FOR AN AP.

ENVIRONMENT DIVISION.
LIBRARY NAME:  VAX

MACRO NAME:  CTOE10

PARAMETER:  P1 - P2

STARTP1.

   MOVE "R" TO DISPOSITION.
   CALL "OPNfil" USING FCB-ES-TEMP,
   RET-STATUS,
   CDMESRES,
   DISPOSITION,
   ES-RECORD-LENGTH,
   NUMBER-OF-RECORDS.

   IF RET-STATUS NOT = KES-FILE-OK
      STRING "CDMESRES OPEN ERROR: " RET-STATUS
      DELIMITED BY SIZE INTO MESG-DESC
      PERFORM PROCESS-ERROR
      GO TO EXIT-PROGRAM.

READP1.

   CALL "INPFIL" USING FCB-ES-TEMP,
   RET-STATUS,
   WS-ES-REC,
   WS-ES-BUFFER-LENGTH,
   RETURN-LENGTH.

   IF RET-STATUS NOT = KES-FILE-OK AND
      RET-STATUS NOT = KES-END-OF-FILE-INPUT
      STRING "WS-ES-REC READ ERROR: " RET-STATUS
      DELIMITED BY SIZE INTO MESG-DESC
      PERFORM PROCESS-ERROR
      GO TO EXIT-PROGRAM.

   IF RET-STATUS = KES-END-OF-FILE-INPUT
      MOVE WS-COUNT TO ES-VAR-P2-P1
      MOVE ZERO TO WS-COUNT
      MOVE ZERO TO ES-NULL-FLAG-P1
      GO TO READP1-EXIT.

   IF WS-NULL-FLAG-P1 NOT = 1
      ADD 1 TO WS-COUNT.
   GO TO READP1.

READP1-EXIT.

   EXIT.

CONTP1.

   MOVE "K" TO DISPOSITION.
   CALL "CLSFIL" USING FCB-ES-TEMP,
   RET-STATUS,
   DISPOSITION.

   IF RET-STATUS NOT = KES-SUCCESSFUL
      STRING "RESULTS FILE CLOSE ERROR: " RET-STATUS
      DELIMITED BY SIZE INTO MESG-DESC
      PERFORM PROCESS-ERROR

   18-29
GO TO EXIT-PROGRAM.

*******************************************************************************
LIBRARY NAME: VAX
MACRO NAME: CTOE11
PARAMETER: P1 - P2

STARTP1.
MOVE "R" TO DISPOSITION.
CALL "OPNFIL" USING FCB-ES-TEMP,
    RET-STATUS,
    CDMESRES,
    DISPOSITION,
    ES-RECORD-LENGTH,
    NUMBER-OF-RECORDS.
    
    IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR OPENING FILE CDMESRES" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
    
    MOVE 1 TO ES-NULL-FLAG-P1.
READP1.
    CALL "INPFIL" USING FCB-ES-TEMP,
        RET-STATUS,
        WS-ES-REC,
        WS-ES-BUFFER-LENGTH,
        RETURN-LENGTH.
        
    IF RET-STATUS = KES-END-OF-FILE-INPUT
    MOVE WS-SUM TO ES-VAR-P2-P1
    MOVE ZERO TO WS-SUM
    GO TO READP1-EXIT.
    
    IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR READING FILE CDMESRES" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
    
    IF WS-NULL-FLAG-P1 NOT = 1
    ADD WS-VAR-P2-P1 TO WS-SUM
    MOVE ZERO TO ES-NULL-FLAG-P1.
    GO TO READP1.
READP1-EXIT.
EXIT.

CONTPL.
MOVE "K" TO DISPOSITION.
CALL "CLSFIL" USING FCB-ES-TEMP,
    RET-STATUS,
    DISPOSITION.
    
    IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR CLOSING FILE CDMESRES" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.

******************************************************************************
LIBRARY NAME: VAX
MACRO NAME: CTOE12
PARAMETER: P1 - P2

STARTP1.
  MOVE "R" TO DISPOSITION.
  CALL "OPNFIL" USING FCB-ES-TEMP,
       RET-STATUS,
       CDMESRES,
       DISPOSITION,
       ES-RECORD-LENGTH,
       NUMBER-OF-RECORDS.
  IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR OPENING FILE CDMESRES" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
  MOVE ZERO TO ES-NULL-FLAG-P1.

READP1.
  CALL "INPFIL" USING FCB-ES-TEMP,
       RET-STATUS,
       WS-ES-REC,
       WS-ES-BUFFER-LENGTH,
       RETURN-LENGTH.
  IF RET-STATUS = KES-END-OF-FILE-INPUT
    PERFORM STARTP1-ZCHK
    COMPUTE ES-VAR-P2-P1 = WS-SUM / WS-COUNT
    MOVE ZERO TO WS-SUM, WS-COUNT
    GO TO READP1-EXIT.
  IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR READING FILE CDMESRES" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
  IF WS-NULL-FLAG-P1 NOT = 1
    ADD 1 TO WS-COUNT
    ADD WS-VAR-P2-P1 TO WS-SUM
    GO TO READP1.

STARTP1-ZCHK.
  IF WS-COUNT = ZERO
    MOVE 1 TO WS-COUNT
    MOVE 1 TO ES-NULL-FLAG-P1.

READP1-EXIT.
  EXIT.

CONTP1.
  MOVE "K" TO DISPOSITION.
  CALL "CLSFIL" USING FCB-ES-TEMP,
       RET-STATUS,
       DISPOSITION.
  IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR CLOSING FILE CDMESRES" TO MESG-DESC
    PERFORM PROCESS-ERROR

18-32
GO TO EXIT-PROGRAM.

**********************************************************
LIBRARY NAME: VAX
MACRO NAME: CTOE13
PARAMETER: P1 - P2 - P3 - P4 - P5 - P6

STARTP1.
  MOVE "R" TO DISPOSITION.
  CALL "OPNFIL" USING FCB-ES-TEMP,
      RET-STATUS,
      CDMESRES,
      DISPOSITION,
      ES-RECORD-LENGTH,
      NUMBER-OF-RECORDS.

  IF RET-STATUS NOT = KES-FILE-OK
     MOVE "ERROR OPENING FILE CDMESRES" TO MESG-DESC
     PERFORM PROCESS-ERROR
     GO TO EXIT-PROGRAM.

  MOVE P4 TO WS-COMP-P6
  MOVE 1 TO ES-NULL-FLAG-P1.

READP1.
  CALL "INPFIL" USING FCB-ES-TEMP,
      RET-STATUS,
      WS-ES-REC,
      WS-ES-BUFFER-LENGTH,
      RETURN-LENGTH.

  IF RET-STATUS = KES-END-OF-FILE-INPUT
     MOVE WS-COMP-VAR3 TO ES-VAR-P2-P1
     GO TO READP1-EXIT.

  IF RET-STATUS NOT = KES-FILE-OK
     MOVE "ERROR READING FILE CDMESRES" TO MESG-DESC
     PERFORM PROCESS-ERROR
     GO TO EXIT-PROGRAM.

  IF WS-NULL-FLAG-P1 NOT = 1 AND
     WS-VAR-P2-P1 P5 THAN WS-COMP-VARP3
     MOVE ZERO TO ES-NULL-FLAG-P1
     MOVE WS-VAR-P2-P1 TO WS-COMP-VARP3.
     GO TO READP1.

READP1-EXIT.

EXIT.

CONTP1.
  MOVE "K" TO DISPOSITION.
  CALL "CLSFIL" USING FCB-ES-TEMP,
      RET-STATUS,
      DISPOSITION.

  IF RET-STATUS NOT = KES-FILE-OK
     MOVE "ERROR CLOSING FILE CDMESRES" TO MESG-DESC
     PERFORM PROCESS-ERROR
     GO TO EXIT-PROGRAM.

*******************************************************************************

18-34
LIBRARY NAME: VAX

MACRO NAME: CTOE14

PARAMETER: P1

P1

EXIT PROGRAM.

EXIT-PROGRAM.

MOVE 1 TO EOF-FLAG.

P1

MOVE SPACES TO ES-REC.

PERFORM DEL-PARA.

EXIT PROGRAM.

COPY ERRPRO OF IISSCLIB.

DEL-PARA.

CALL "DELFIL" USING MY-HOST

CDMCSRES.
LIBRARY NAME: VAX

MACRO NAME: CTOE15

PARAMETER:

CK-DISTINCT.
ADD 1 TO DSUB.
IF DSUB GREATER THAN 1000
MOVE "OVERFLOW OF UNIQUE VALUES TABLE" TO MESG-DESC
MOVE KES-TABLE-OVERFLOW TO RET-STATUS
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
IF DISTINCT-ENTRY (DSUB) EQUAL SPACES
MOVE WS-COMP-VAR TO DISTINCT-VAR (DSUB)
call "OUTFIL" USING FCB-ES-TEMP,
RET-STATUS,
WS-ES-REC,
ES-RECORD-LENGTH
IF RET-STATUS = KES-FILE-OK
GO TO CK-DISTINCT-EXIT
ELSE
MOVE "ERROR WRITING TO FILE CDMESRES" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
IF DISTINCT-VAR (DSUB) EQUAL WS-COMP-VAR
GO TO CK-DISTINCT-EXIT.
GO TO CK-DISTINCT.
CK-DISTINCT-EXIT.
EXIT.

INITIALIZE-TABLE.
ADD 1 TO DSUB.
IF DSUB GREATER THAN 1000
MOVE ZERO TO DSUB
GO TO INITIALIZE-TABLE-EXIT.
MOVE SPACES TO DISTINCT-VAR (DSUB).
GO TO INITIALIZE-TABLE.
INITIALIZE-TABLE-EXIT.
EXIT.
LIBRARY NAME: VAX
MACRO NAME: CTOE2

PARAMETER: P1

DATA DIVISION.
WORKING-STORAGE SECTION.
01 ES-TEMP-REC PIC X(P1).
01 ES-RECORD-LENGTH PIC S9(9) COMP VALUE P1.

* 01 CS-REC.
  03 CS-NULL-FLAGS.

******************************************************************************
LIBRARY NAME: VAX
MACRO NAME: CTOE3

PARAMETER:
***********************************************************************************************************************************************
* PART OF WORKING STORAGE WHEN FUNCTIONS PERFORMED NONE OF *
* WHICH HAVE DISTINCT APPLIED.
***********************************************************************************************************************************************
*
01 WS-COUNT PIC S9(9) VALUE ZERO.
01 WS-SUM PIC S9(9)V9(9) VALUE ZERO.
01 WS-COMP-VAR.
  03 WS-COMP-VARX PIC X(P1) VALUE SPACES.
  03 WS-COMP-VARN REDEFINES WS-COMP-VARX PIC S9(9)V9(9).

18-38
LIBRARY NAME: VAX

MACRO NAME: CTOE4

PARAMETER: P1 - P2 - P3

*

******************************************************************************
*  COMMON WORKING STORAGE FOR ALL CASES
******************************************************************************
*

01 CDMCSRES PIC X(80) VALUE SPACES.
01 CDMESRES PIC X(80) VALUE SPACES.
01 CDMTMPFL PIC X(80) VALUE SPACES.
01 MY-HOST PIC XXX VALUE "P1".
01 MESG-DESC PIC X(60) VALUE SPACES.
01 MODULE-NAME PIC X(10) VALUE "P2".
01 FIRST-RECORD PIC S9(9) COMP.
01 FCB-ES-TEMP PIC S9(9) COMP.
01 FCB-TEMP-FILE PIC S9(9) COMP.
01 FCB-CS-INPUT PIC S9(9) COMP.
01 CS-RECORD-LENGTH PIC S9(9) COMP.
01 WS-ES-BUFFER-LENGTH PIC S9(9) COMP VALUE P3.
01 DISPOSITION PIC X.
01 NUMBER-OF-RECORDS PIC S9(9) COMP VALUE 2000.
01 RETURN-LENGTH PIC S9(9) COMP.
01 TEMP-RECORD-LENGTH PIC S9(9) COMP.
COPY CHKCDM OF IISSCLIB.
COPY ERRCDM OF IISSCLIB.
COPY ERRFS OF IISSCLIB.
LIBRARY NAME: VAX

MACRO NAME: CTOE4A

PARAMETER: P1

* PART OF WORKING STORAGE SECTION ADDED
* WHEN DISTINCT PROCESS AND FUNCTION PERFORMED
* ON VARIABLES.

*---------------------------------------------------------------*
01 WS-COMP-VAR.
  03 WS-COMP-NULL-FLAG PIC 9.
  03 WS-COMP-VARX PIC X(P1) VALUE SPACES.
  03 WS-COMP-VARN REDEFINES WS-COMP-VARX PIC S9(9)V9(9).
01 DSUB PIC S9999 VALUE ZERO.
01 DISTINCT-TABLE.
  03 DISTINCT-ENTRY OCCURS 1000 TIMES.
    05 DISTINCT-VAR.
      07 FILLER PIC 9.
      07 FILLER PIC X(P1)
01 WS-COUNT PIC S9(9) VALUE ZERO.
01 WS-SUM PIC S9(9)V9(9) VALUE ZERO.

C*
LIBRARY NAME: VAX

MACRO NAME: CTOE4B

PARAMETER: P1 - P2

* FIRST PART OF KEY AND FILE INFORMATION USED BY "NISSORT"
* TO CREATE SORT-KEY AND SUBSEQUENTLY SORT FILE. THIS MACRO
* IS ALWAYS FOLLOWED BY CTOE4C WHICH CONTAINS EXPLICIT VALUES

01 INPUT-FILE-P1.
   03 FILE-NAME-P1 PIC X(80) VALUE SPACES.
   03 FILE-REC-KEY-USED PIC 9(6) COMP VALUE P2.
   03 FILLER PIC 9(6) COMP VALUE ZERO.

*****************************************************************************
LIBRARY NAME: VAX

MACRO NAME: CTOE4C

PARAMETER: P1 - P2 - P3 - P4

*SECOND PART OF KEY AND FILE INFORMATION USED BY "NISSORT"
*CONTAINS EXPLICIT VALUES.

   03 FILLER PIC 9(6) COMP.
   03 FILLER PIC S9(6) COMP.
   03 FILLER PIC S9(6) COMP VALUE P1.
   03 FILLER PIC X VALUE "P3".
   03 FILLER PIC S99 COMP.
   03 FILLER PIC X VALUE "P4".

******************************************************************************
LIBRARY NAME: VAX
MACRO NAME: CTOE5

PARAMETER:
* LINKAGE SECTION FOR ALL CASES
* LINKAGE SECTION.
* INPUT ARGUMENTS.
  01 CALL-FLAG PIC 9.
  * EQUAL TO 1 IF FIRST TIME PROGRAM CALLED; 2 IF 2-?
  * TIMES PROGRAM CALLED OR 3 IF PROGRAM IS TO QUIT
  * EARLY
  01 CDM-CS-RESULTS-FILE PIC X(80).
  * FILE NAME OF INPUTS TO CS-ES-RTH.
  * OUTPUT ARGUMENTS
  01 EOF-FLAG PIC 9.
  * SET TO 1 IF NO MORE ES RECORDS TO BE SENT TO AP
  01 RET-STATUS PIC X(5).
  01 ES-NULL-FLAGS.
    03 ES-NULL-FLAG-01 PIC 9.
    03 ES-NULL-FLAG-02 PIC 9.
    03 ES-NULL-FLAG-03 PIC 9.
    03 ES-NULL-FLAG-04 PIC 9.
    03 ES-NULL-FLAG-05 PIC 9.
    03 ES-NULL-FLAG-06 PIC 9.
    03 ES-NULL-FLAG-07 PIC 9.
    03 ES-NULL-FLAG-08 PIC 9.
    03 ES-NULL-FLAG-09 PIC 9.
    03 ES-NULL-FLAG-10 PIC 9.
    03 ES-NULL-FLAG-11 PIC 9.
    03 ES-NULL-FLAG-12 PIC 9.
    03 ES-NULL-FLAG-13 PIC 9.
    03 ES-NULL-FLAG-14 PIC 9.
    03 ES-NULL-FLAG-15 PIC 9.
    03 ES-NULL-FLAG-16 PIC 9.
    03 ES-NULL-FLAG-17 PIC 9.
    03 ES-NULL-FLAG-18 PIC 9.
    03 ES-NULL-FLAG-19 PIC 9.
    03 ES-NULL-FLAG-20 PIC 9.
    03 ES-NULL-FLAG-21 PIC 9.
    03 ES-NULL-FLAG-22 PIC 9.

18-43
03 ES-NUL-FLAG-23 PIC 9.
03 ES-NUL-FLAG-24 PIC 9.
03 ES-NUL-FLAG-25 PIC 9.
01 ES-REC.

***********************************************************************
LIBRARY NAME: VAX
MACRO NAME: CTOE5A

PARAMETER:
* THIS MACRO WRITES THE TEMPORARY ES FILE
* 
  MOVE WS-ES-REC TO ES-TEMP-REC.
  CALL "OUTFIL" USING FCB-ES-TEMP,
       RET-STATUS,
       ES-TEMP-REC,
       ES-RECORD-LENGTH.
  IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR WRITING TO FILE CDMESRES" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
    GO TO CS-ES-RTN.
CS-ES-RTN-EXIT.
EXIT.

******************************************************************************
LIBRARY NAME: VAX
MACRO NAME: CTOE6

PARAMETER:
* BEGINNING OF PROCEDURE DIVISION FOR CASE 2 SELECT
* CERTAIN FIELDS - NO FUNCTIONS, DISTINCTS OR ORDER
* BY.

PROCEDURE DIVISION USING CALL-FLAG,
CDM-CS-RESULTS-FILE,
CS-QUALIFY-VAR,
ES-NULL-FLAGS,
ES-REC,
EOF-FLAG,
RET-STATUS.

START-PROGRAM.
MOVE SPACES TO ES-REC.
MOVE ZERO TO EOF-FLAG.
MOVE KES-SUCCESSFUL TO RET-STATUS.
MOVE CDM-CS-RESULTS-FILE TO CDMCSRES.
IF CALL-FLAG = 3
  MOVE "K" TO DISPOSITION
  CALL "CLSFIL" USING FCB-CS-INPUT,
  RET-STATUS,
  DISPOSITION
  IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR CLOSING FILE CDMCSRES" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM
  ELSE
    GO TO EXIT-PROGRAM.
ENDIF
IF CALL-FLAG = 1
  MOVE "R" TO DISPOSITION
  CALL "OPNFIL" USING FCB-CS-INPUT,
  RET-STATUS,
  CDMCSRES,
  DISPOSITION,
  CS-RECORD-LENGTH,
  NUMBER-OF-RECORDS
  IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR OPENING FILE CDMCSRES" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM
  ELSE
    GO TO EXIT-PROGRAM.
ENDIF
CS-ES-RTN.

18-46
CALL "INPFIL" USING FCB-CS-INPUT,
    RET-STATUS,
    CS-REC,
    CS-RECORD-LENGTH,
    RETURN-LENGTH.
IF RET-STATUS = KES-END-OF-FILE-INPUT
    MOVE "K" TO DISPOSITION
    CALL "CLSFIL" USING FCB-CS-INPUT,
    RET-STATUS,
    DISPOSITION
IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR CLOSING FILE CDMCSRES"
    TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM
ELSE
    GO TO EXIT-PROGRAM.
IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR READING FILE CDMCSRES" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
LIBRARY NAME: VAX

MACRO NAME: CTOE6A

PARAMETER:
* BEGINNING OF PROCEDURE DIVISION FOR CASE3 - FUNCTIONS OR
* FUNCTION DISTINCTS. 1 OUTPUT RECORD.
* PROCEDURE DIVISION USING CALL-FLAG,
  CDM-CS-RESULTS-FILE,
  CS-QUALIFY-VAR,
  ES-NULL-FLAGS,
  ES-REC,
  EOF-FLAG,
  RET-STATUS.

START-PROGRAM.
   MOVE CDM-CS-RESULTS-FILE TO CDMCSRES.
   MOVE SPACES TO ES-REC.
   MOVE ZERO TO EOF-FLAG.
   MOVE KES-SUCCESSFUL TO RET-STATUS.
   IF CALL-FLAG = 3
      GO TO EXIT-PROGRAM.
   IF CALL-FLAG > 1
      CALL "NAMFIL" USING CDMESRES.
      IF CDMESRES = LOW-VALUE
         MOVE "TRYING TO GET TEMPORARY FILE NAME1" TO MESG-DESC
         PERFORM PROCESS-ERROR
         GO TO EXIT-PROGRAM.
      CALL "NAMFIL" USING CDMTMPFL.
      IF CDMTMPFL = LOW-VALUE
         MOVE "TRYING TO GET TEMPORARY FILE NAME2" TO MESG-DESC
         PERFORM PROCESS-ERROR
         GO TO EXIT-PROGRAM.
   MOVE "R" TO DISPOSITION.
   CALL "OPNFIL" USING FCB-CS-INPUT,
      RET-STATUS,
      CDMCSRES,
      DISPOSITION,
      CS-RECORD-LENGTH,
      NUMBER-OF-RECORDS.
   IF RET-STATUS NOT = KES-FILE-OK
      MOVE "ERROR OPENING FILE CDMCSRES" TO MESG-DESC
      PERFORM PROCESS-ERROR
      GO TO EXIT-PROGRAM.

18-48
MOVE "W" TO DISPOSITION.
CALL "OPNFIL" USING FCB-ES-TEMP,
    RET-STATUS,
    CDMESRES,
    DISPOSITION,
    ES-RECORD-LENGTH,
    NUMBER-OF-RECORDS.

IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR OPENING FILE CDMESRES" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.

CS-ES-RTN.
    CALL "INPFIL USING FCB-CS-INPUT,
        RET-STATUS,
        CS-REC,
        CS-RECORD-LENGTH,
        RETURN-LENGTH.

IF RET-STATUS = KES-END-OF-FILE-INPUT
    MOVE "K" TO DISPOSITION
    CALL "CLSFIL" USING FCB-CS-INPUT,
        RET-STATUS,
        DISPOSITION

    IF RET-STATUS NOT = KES-FILE-OK
        MOVE "ERROR CLOSING FILE CDMCSRES" TO MESG-DESC
        PERFORM PROCESS-ERROR
        GO TO EXIT-PROGRAM
    ELSE
        CALL "CLSFIL" USNG FCB-ES-TEMP,
            RET-STATUS,
            DISPOSITION

        IF RET-STATUS NOT = KES-FILE-OK
            MOVE "ERROR CLOSING FILE CDMESRES"
                TO MESG-DESC
            PERFORM PROCESS-ERROR
        IF RET-STATUS NOT = KES-FILE-OK
            GO TO EXIT-PROGRAM
        MOVE "ERROR READING FILE CDMCSRES" TO MMESG-DESC
        ELSE
            PERFORM PROCESS-ERROR
            GO TO EXIT-PROGRAM.
        GO TO CS-ES-RTN-EXIT.
LIBRARY NAME: VAX
MACRO NAME: CTOE6B

PARAMETER:

* BEGINNING OF PROCEDURE DIVISION FOR CASE2 WHERE
* ONE SORT IS REQUIRED.
*
PROCEDURE DIVISION USING CALL-FLAG,
CDM-CS-RESULTS-FILE,
CS-QUALIFY-VAR,
ES-NULL-FLAGS,
ES-REC,
EOF-FLAG,
RET-STATUS.

START-PROGRAM.
MOVE SPACES TO ES-REC.
MOVE ZERO TO EOF-FLAG.
MOVE KES-SUCCESSFUL TO RET-STATUS.
MOVE CDM-CS-RESULTS-FILE TO CDMCSRES.
IF CALL-FLAG = 3
  MOVE "K" TO DISPOSITION
  CALL "CLSFIL" USING FCB-TEMP-FILE,
                    RET-STATUS,
                    DISPOSITION
  IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR CLOSING FILE CDMTMPFL"
    TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM
  ELSE
    GO TO EXIT-PROGRAM.
ELSE
  GO TO EXIT-PROGRAM.
IF CALL-FLAG NOT EQUAL 1
  GO TO RELEASE-RECORDS.
CALL "NAMFIL" USING CDMESRES.
IF CDMESRES = LOW-VALUE
  MOVE "TRYING TO GET TEMORARY FILE-NAME1"
  TO MESG-DESC
  PERFORM PROCESS-ERROR
  GO TO EXIT-PROGRAM.
CALL "NAMFIL" USING CDMTMPFL.
IF CDMTMPFL = LOW-VALUE
  MOVE "TRYING TO GET TEMORARY FILE-NAME2"
  TO MESG-DESC
  PERFORM PROCESS-ERROR

18-50
GO TO EXIT-PROGRAM.
MOVE "R" TO DISPOSITION.
CALL "OPNFIL" USING FCB-CS-INPUT,
   RET-STATUS,
   CDMCSRES,
   DISPOSITION,
   CS-RECORD-LENGTH,
   NUMBER-OF-RECORDS.

IF RET-STATUS NOT = KES-FILE-OK
   MOVE "ERROR OPENING FILE CDMCSRES" TO MESS-DESC
   PERFORM PROCESS-ERROR
   GO TO EXIT-PROGRAM.
MOVE "W" TO DISPOSITION.
CALL "OPNFIL" USING FCB-ES-TEMP,
   RET-STATUS,
   CDMESRES,
   DISPOSITION,
   ES-RECORD-LENGTH,
   NUMBER-OF-RECORDS.

IF RET-STATUS NOT = KES-FILE-OK
   MOVE "ERROR OPENING FILE CDMESRES" TO MESS-DESC
   PERFORM PROCESS-ERROR
   GO TO EXIT-PROGRAM.
CS-ES-RTN.
CALL "INPFIL" USING FCB-CS-INPUT,
   RET-STATUS,
   CS-REC,
   CS-RECORD-LENGTH,
   RETURN-LENGTH.

IF RET-STATUS = KES-END-OF-FILE-INPUT
   MOVE "K" TO DISPOSITION
   CALL "CLSFIL" USING FCB-CS-INPUT,
   RET-STATUS,
   DISPOSITION
   IF RET-STATUS NOT = KES-FILE-OK
      MOVE "ERROR CLOSING FILE CDMCSRES" TO MESS-DESC
      PERFORM PROCESS-ERROR
      GO TO EXIT-PROGRAM
   ELSE
      CALL "CLSFIL" USING FCB-ES-TEMP,
      RET-STATUS,
      DISPOSITION
   IF RET-STATUS NOT = KES-FILE-OK
      MOVE "ERROR CLOSING FILE CDMESRES" TO MESS-DESC
      PERFORM PROCESS-ERROR
      GO TO EXIT-PROGRAM
   ELSE
      GO TO CS-ES-RTN-EXIT.
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR READING FILE CDMCSREC" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.

*****************************************************************************
LIBRARY NAME: VAX

MACRO NAME: CTOE6B1

PARAMETER:

START-SORT.

MOVE CDMESRES TO FILE-NAME-1.
CALL "CDMPSOR" USING INPUT-FILE-1,
CDMTPFIL,

*  

MESG-DESC,
RET-STATUS.

MOVE RET-STATUS TO QCS-CDMP-CHECK-STATUS.
IF NOT QCS-SUCCESSFUL
MOVE "SORT/MERGE PROGRAM FAILED" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE "R" TO DISPOSITION.
CALL "OPNFIL" USING FCB-TEMP-FILE,
RET-STATUS,
CDMTPFIL,
DISPOSITION,
TEMP-RECORD-LENGTH,
NUMBER-OF-RECORDS.

IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR OPENING FILE CDMTPFIL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE SPACES TO DST-REC.
RELEASE-RECORDS.
CALL "INPFIL" USING FCB-TEMP-FILE,
RET-STATUS,
WS-ES-REC,
WS-ES-BUFFER-LENGTH,
RETURN-LENGTH.

IF RET-STATUS = KES-END-OF-FILE-INPUT
MOVE "K" TO DISPOSITION
CALL "CLSFIL" USING FCB-TEMP-FILE,
RET-STATUS,
DISPOSITION

IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR CLOSING FILE CDMTPFIL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
ELSE
GO TO EXIT-PROGRAM.

IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR READING FILE CDMTPFIL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE 0 TO EOF-FLAG.

*******************************************************************************
LIBRARY NAME: VAX
MACRO NAME: CTOE6C

PARAMETER

* BEGINNING OF PROCEDURE DIVISION FOR CASE1 WHEN IT
* REQUIRES 2 SORTS. (DISTINCT PROCESS AND ORDER BY WHERE
* ALL ORDER BY VARIABLES AREN'T PROJECTED)

PROCEDURE DIVISION USING CALL-FLAG,
CDM-CS-RESULTS-FILE,
CS-QUALIFY-VAR,
ES-NULL-FLAGS,
ES-REC,
EOF-FLAG,
RET-STATUS.

START-PROGRAM.
  MOVE SPACES TO ES-REC.
  MOVE ZERO TO EOF-FLAG.
  MOVE KES-SUCCESSFUL TO RET-STATUS.
  MOVE CDM-CS-RESULTS-FILE TO CDMCSRES.
  IF CALL-FLAG EQUAL 3
    MOVE "K" TO DISPOSITION
    CALL "CLSFIL" USING FCB-TEMP-FLE.
    RET-STATUS,
    DISPOSITION
    IF RET-STATUS NOT = KES-FILE-OK
      MOVE "ERROR CLOSING FILE CDMTMPFL"
      TO MESG-DESC
      PERFORM PROCESS-ERROR
      GO TO EXIT-PROGRAM
    ELSE
      GO TO EXIT-PROGRAM.
  ELSE
    GO TO EXIT-PROGRAM.
  IF CALL-FLAG NOT EQUAL 1
    GO TO RELEASE-SORT-RECS.
    CALL "NAMFIL" USING CDMESRES.
    IF CDMESRES = LOW-VALUE
      MOVE "TRYING TO GET TEMPORARY FILE-NAME1"
      TO MESG-DESC
      PERFORM PROCESS-ERROR
      GO TO EXIT-PROGRAM.
    CALL "NAMFIL" USING CDTMPFL.
    IF CDTMPFL = LOW-VALUE
      MOVE "TRYING TO GET TEMPORARY FILE-NAME2"
      TO MESG-DESC
      PERFORM PROCESS-ERROR
      GO TO EXIT-PROGRAM.

18-55
MOVE "R" TO DISPOSITION.
CALL "OPNFIL" USING FCB-CS-INPUT,
       RET-STATUS,
       CDMCSRES,
       DISPOSITION,
       CS-RECORD-LENGTH,
       NUMBER-OF-RECORDS.

IF RET-STATUS NOT = KES-FILE-OK
   MOVE "ERROR OPENING FILE CDMCSRES" TO MESG-DESC
   PERFORM PROCESS-ERROR
   GO TO EXIT-PROGRAM.

MOVE "W" TO DISPOSITION.
CALL "OPNFIL" USING FCB-ES-TEMP,
       RET-STATUS,
       CDMESRES,
       DISPOSITION,
       ES-RECORD-LENGTH,
       NUMBER-OF-RECORDS.

IF RET-STATUS NOT = KES-FILE-OK
   MOVE "ERROR OPENING FILE CDMESRES" TO MESG-DESC
   PERFORM PROCESS-ERROR
   GO TO EXIT-PROGRAM.

CS-ES-RTN.

CALL "INPFIL" USING FB-CS-INPUT,
       RET-STATUS,
       CS-REC,
       CS-RECORD-LENGTH,
       RETURN-LENGTH.

IF RET-STATUS = KES-END-OF-FILE-INPUT
   MOVE "K" TO DISPOSITION
   CALL "CLSFIL" USING FCB-CS-INPUT,
          RET-STATUS,
          DISPOSITION
   IF RET-STATUS NOT = KES-FILE-OK
      MOVE "ERROR IN CLOSING FILE CDMCSRES" TO MESG-DESC
      PERFORM PROCESS-ERROR
      GO TO EXIT-PROGRAM
   ELSE
      CALL "CLSFIL" USING FCB-ES-TEMP,
             RET-STATUS,
             DISPOSITION
      IF RET-STATUS NOT = KES-FILE-OK
         MOVE "ERROR IN CLOSING FILE CDMESRES" TO MESG-DESC
         PERFORM PROCESS-ERROR
         GO TO EXIT-PROGRAM
      ELSE
         GO TO CS-ES-RTN-EXIT.
IF RET-STATUS NOT = KES-FILE-OK
  MOVE "ERROR IN READING FILE CDMCSRES" TO MESG-DESC
  PERFORM PROCESS-ERROR
  GO TO EXIT-PROGRAM.
LIBRARY NAME: VAX

MACRO NAME: CTOE6C1

PARAMETER:
SORT01.

MOVE CDMESRES TO FILE-NAME-1
CALL "CDMPSOR" USING INPUT-FILE-1
CDMTMPFL,

* MESG-DESC,
RET-STATUS.
MOVE RET-STATUS TO QCS-CDMP-CHECK-STATUS.
IF NOT QCS-SUCCESSFUL
MOVE "SORT/MERGE PROGRAM FAILED"
TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE "R" TO DISPOSITION.
CALL "OPNFIL" USING FCB-TEMP-FILE,
RET-STATUS,
CDMTMPFL,
DISPOSITION,
TEMP-RECORD-LENGTH,
NUMBER-OF-RECORDS.
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR OPENING FILE CDMTMPFL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE "W" TO DISPOSITION.
CALL "OPNFIL" USING FCB-ES-TEMP,
RET-STATUS,
CDMESRES,
DISPOSITION,
ES-RECORD-LENGTH,
NUMBER-OF-RECORDS.
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR OPENING FILE CDMEESRES" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE ZERO TO FIRST-RECORD.
RELEASE-RECORDS.
CALL "INPFIL" USING FCB-TEMP-FILE,
RET-STATUS,
WS-ES-REC,
WS-ES-BUFFER-LENGTH,
RETURN-LENGTH.
IF RET-STATUS = KES-END-OF-FILE-INPUT
MOVE "K" TO DISPOSITION

18-53
CALL "CLSFIL" USING FCB-TEMP-FILE,
RET-STATUS,
DISPOSITION
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR CLOSING FILE CDMTMPFL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM
ELSE
CALL "CLSFIL" USING FCB-ES-TEMP,
RET-STATUS,
DISPOSITION
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR CLOSING FILE CDMESRES"
TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM
ELSE
GO TO SECOND-SORT.
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR READING FILE CDMTMPFL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE 0 TO EOF-FLAG.
ADD 1 TO FIRST-RECORD.
*****************************************************************************
LIBRARY-NAME: VAX

MACRO NAME: CTOE6D

PARAMETER:
SECOND-SORT.

MOVE CDMESRES TO FILE-NAME-2.
CALL "CDMPSOR" USING INPUT-FILE-2
CDMTMPFL

* MESG-DESC
RET-STATUS.

MOVE RET-STATUS TO QCS-CDMP-CHECK-STATUS.
IF NOT QCS-SUCCESSFUL
MOVE "SORT/MERGE PROGRAM FAILED" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE "R" TO DISPOSITION.
CALL "OPNFIL" USING FCB-TEMP-FILE,
RET-STATUS,
CDMTMPFL,
DISPOSITION,
TEMP-RECORD-LENGTH,
NUMBER-OF-RECORDS.

IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR OPENING FILE CDMTMPFL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.

RELEASE-SORT-RECS.
CALL "INPFIL" USING FCB-TEMP-FILE,
RET-STATUS,
WS-ES-REC,
WS-ES-BUFFER-LENGTH,
RETURN-LENGTH.

IF RET-STATUS = KES-END-OF-FILE-INPUT
MOVE "K" TO DISPOSITION
CALL "CLSFIL" USING FCB-TEMP-FILE,
RET-STATUS,
DISPOSITION

IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR CLOSING FILE CDMTMPFL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM
ELSE
GO TO EXIT-PROGRAM.

IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR READING FILE CDMTMPFL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE 0 TO EOF-FLAG.

*******************************************************************************
LIBRARY NAME: VAX
MACRO NAME: CTOE7
PARAMETER: P1 - P2
STARTP1.
MOVE "R" TO DISPOSITION.
CALL "OPNFIL" USING FCB-ES-TEMP,
RET-STATUS,
CDMESRES,
DISPOSITION,
ES-RECORD-LENGTH,
NUMBER-OF-RECORDS.
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR OPENING FILE CDMESRES" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE "W" TO DISPOSITION.
CALL "OPNFIL" USING FCB-TEMP-FILE,
RET-STATUS,
CDMTMPFL,
DISPOSITION,
TEMP-RECORD-LENGTH,
NUMBER-OF-RECORDS.
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR OPENING FILE CDMTMPFL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE ZERO TO DSUB.
PERFORM INITIALIZE-TABLE THRU INITIALIZE-TABLE-EXIT.
READP1.
CALL "INPFIL" USING FCB-ES-TEMP,
RET-STATUS,
WS-ES-REC,
WS-ES-BUFFER-LENGTH,
RETURN-LENGTH.
IF RET-STATUS = KES-END-OF-FILE-INPUT
GO TO READP1-EXIT.
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR READING FILE CDMESRES" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE WS-VAR-P2-P1 TO WS-COMP-VARP3.
MOVE WS-NULL-FLAG-P1 TO WS-COMP-NULL-FLAG.
MOVE ZERO TO DSUB.
PERFORM CK-DISTINCT THRU CK-DISTINCT-EXIT.
GO TO READP1.
READP1-EXIT.
  EXIT.
CONTPI.
  MOVE "K" TO DISPOSITION.
  CALL "CLSFIL" USING FCB-TEMP-FILE,
    RET-STATUS,
    DISPOSITION.

  IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR CLOSING FILE CDMTMPFL" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
  MOVE "R" TO DISPOSITION.
  CALL "OPNFIL" USING FCB-TEMP-FILE,
    RET-STATUS,
    CDMTMPFL,
    DISPOSITION,
    TEMP-RECORD-LENGTH,
    NUMBER-OF-RECORDS.

  IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR OPENING FILE CDMTMPFL" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
READ-TEMPP1.
  CALL "INPFIL" USING FCB-TEMP-FILE,
    RET-STATUS,
    WS-ES-REC,
    WS-ES-BUFFER-LENGTH,
    RETURN-LENGTH.

  IF RET-STATUS = KES-END-OF-FILE-INPUT
    MOVE WS-COUNT TO ES-VAR-P2-P1
    MOVE ZERO TO WS-COUNT
    MOVE ZERO TO ES-NULL-FLAG-P1
    GO TO READ-TEMPP1-EXIT.
  IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR READING FILE CDMTMPFL" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
  IF WS-NULL-FLAG-P1 NOT = 1
    ADD 1 TO WS-COUNT.
  GO TO READ-TEMPP1.
READ-TEMPP1-EXIT.
  EXIT.
CONTPIA.
  MOVE "K" TO DISPOSITION.
  CALL "CLSFIL" USING FCB-TEMP-FILE,
    RET-STATUS,
    DISPOSITION.

  IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR CLOSING FILE CDMTMPFL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
CALL "CLSFIL" USING FCB-ES-TEMP,
   RET-STATUS,
   DISPOSITION.
IF RET-STATUS NOT = KES-FILE-OK
   MOVE "ERROR CLOSING FILE CDMSRES" TO MESG-DESC
   PERFORM PROCESS-ERROR
   GO TO EXIT-PROGRAM.

******************************************************************************
LIBRARY NAME: VAX
MACRO NAME: CTOE8
PARAMETER: P1 - P2
STARTP1.

MOVE "R" TO DISPOSITION.
CALL "OPNFIL" USING FCB-ES-TEMP,
RET-STATUS,
CDMESRES,
DISPOSITION,
ES-RECORD-LENGTH,
NUMBER-OF-RECORDS.

IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR OPENING FILE CDMESRES" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
MOVE "W" TO DISPOSITION.
CALL "OPNFIL" USING FCB-TEMP-FILE,
RET-STATUS,
CDMTMPFL,
DISPOSITION,
TEMP-RECORD-LENGTH,
NUMBER-OF-RECORDS.

IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR OPENING FILE CDMTMPFL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.

MOVE ZERO TO DSUB.
PERFORM INITIALIZE-TABLE THRU INITIALIZE-TABLE-EXIT.
READP1.

CALL "INPFIL" USING FCB-ES-TEMP,
RET-STATUS,
WS-ES-REC,
WS-ES-BUFFER-LENGTH,
RETURN-LENGTH.

IF RET-STATUS = KES-END-OF-FILE-INPUT
GO TO READP1-EXIT.
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR READING FILE CDMESRES" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.

MOVE WS-VAR-P2-P1 TO WS-COMP-VARP3.
MOVE WS-NULL-FLAG-P1 TO WS-COMP-NULL-FLAG.
MOVE ZERO TO DSUB.
PERFORM CK-DISTINCT THRU CK-DISTINCT-EXIT.
GO TO READP1.
READP1-EXIT.
EXIT.

18-65
CONTP1.
MOVE "K" TO DISPOSITION.
CALL "CLSFIL" USING FCB-TEMP-FILE,
      RET-STATUS,
      DISPOSITION.
      IF RET-STATUS NOT = KES-FILE-OK
         MOVE "ERROR CLOSING FILE CDMTMPFL" TO MESG-DESC
         PERFORM PROCESS-ERROR
         GO TO EXIT-PROGRAM.
      MOVE "R" TO DISPOSITION.
      CALL "OPNFIL" USING FCB-TEMP-FILE,
            RET-STATUS,
            CDMTMPFL,
            DISPOSITION,
            TEMP-RECORD-LENGTH,
            NUMBER-OF-RECORDS.
      IF RET-STATUS NOT = KES-FILE-OK
         MOVE "ERROR OPENING FILE CDMTMPFL" TO MESG-DESC
         PERFORM PROCESS-ERROR
         GO TO EXIT-PROGRAM.
      MOVE 1 TO ES-NULL-FLAG-P1.
READ-TEMPP1.
      CALL "INPFIL" USING FCB-TEMP-FILE,
            RET-STATUS,
            WS-ES-REC,
            WS-ES-BUFFER-LENGTH,
            RETURN-LENGTH.
      IF RET-STATUS = KES-END-OF-FILE-INPUT
         MOVE WS-SUM TO ES-VAR-P2-P1
      MOVE ZERO TO WS-SUM
      GO TO READ-TEMPP1-EXIT.
      IF RET-STATUS NOT = KES-FILE-OK
         MOVE "ERROR READIN FILE CDMTMPFL" TO MESG-DESC
         PERFORM PROCESS-ERROR
         GO TO EXIT-PROGRAM.
      IF WS-NULL-FLAG-P1 NOT = 1
         ADD WS-VAR-P2-P1 TO WS-SUM
      MOVE ZERO TO ES-NULL-FLAG-P1.
      GO TO READ-TEMPP1.
READ-TEMPP1-EXIT.
EXIT.
CONTP1A.
MOVE "K" TO DISPOSITION.
CALL "CLSFIL" USING FCB-TEMP-FILE,
      RET-STATUS,
      DISPOSITION.
      IF RET-STATUS NOT = KES-FILE-OK
         MOVE "ERROR CLOSING FILE CDMTMPFL" TO MESG-DESC
         PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
CALL "CLSFIL" USING FCB-ES-TEMP,
       RET-STATUS,
       DISPOSITION.
IF RET-STATUS NOT = KES-FILE-OK
   MOVE "ERROR CLOSING FILE CDMESRES" TO MESG-DESC
   PERFORM PROCESS-ERROR
   GO TO EXIT-PROGRAM.
******************************************************************************
LIBRARY NAME: VAX
MACRO NAME: CTOE9
PARAMETER: P1 - P2
STARTP1.
    MOVE "R" TO DISPOSITION.
    CALL "OPNFIL" USING FCB-ES-TEMP,
             RET-STATUS,
             CDMESRES,
             DISPOSITION,
             ES-RECORD-LENGTH,
             NUMBER-OF-RECORDS.
    IF RET-STATUS NOT = KES-FILE-OK
        MOVE "ERROR OPENING FILE CDMESRES" TO MESS-DESC
        PERFORM PROCESS-ERROR
        GO TO EXIT-PROGRAM.
    MOVE "W" TO DISPOSITION.
    CALL "OPNFIL" USING FCB-TEMP-FILE,
             RET-STATUS,
             CDMTMPFL,
             DISPOSITION,
             TEMP-RECORD-LENGTH,
             NUMBER-OF-RECORDS.
    IF RET-STATUS NOT = KES-FILE-OK
        MOVE "ERROR OPENING FILE CDMTMPFL" TO MESS-DESC
        PERFORM PROCESS-ERROR
        GO TO EXIT-PROGRAM.
    MOVE ZERO TO DSUB.
    PERFORM INITIALIZE-TABLE THRU INITIALIZE-TABLE-EXIT.
READP1.
    CALL "INPFIL" USING FCB-ES-TEMP,
             RET-STATUS,
             WS-ES-REC,
             WS-ES-BUFFER-LENGTH,
             RETURN-LENGTH.
    IF RET-STATUS = KES-END-OF-FILE-INPUT
        GO TO READP1-EXIT.
    IF RET-STATUS NOT = KES-FILE-OK
        MOVE "ERROR READING FILE CDMESRES" TO MESS-DESC
        PERFORM PROCESS-ERROR
        GO TO EXIT-PROGRAM.
    MOVE WS-VAR-P2-P1 TO WS-COMP-VARP3.
    MOVE WS-NULL-FLAG-P1 TO WS-CC:NULL-FLAG.
    MOVE ZERO TO DSUB.
    PERFORM CK-DISTINCT THRU CK-DISTINCT-EXIT.
    GO TO READP1.
READP1-EXIT.
EXIT.
CONTPI
    MOVE "K" TO DISPOSITION.
    CALL "CLSFIL" USING FCB-TEMP-FILE,
        RET-STATUS,
        DISPOSITION.
    IF RET-STATUS NOT = KES-FILE-OK
        MOVE "ERROR CLOSING FILE CDMTMPFL" TO MESG-DESC
        PERFORM PROCESS-ERROR
        GO TO EXIT-PROGRAM.
    MOVE "R" TO DISPOSITION.
    CALL "OPNFIL" USING FCB-TEMP-FILE,
        RET-STATUS,
        CDMTMPFL,
        DISPOSITION,
        TEMP-RECORD-LENGTH,
        NUMBER-OF-RECORDS.
    IF RET-STATUS NOT = KES-FILE-OK
        MOVE "ERROR OPENING FILE CDMTMPFL" TO MESG-DESC
        PERFORM PROCESS-ERROR
        GO TO EXIT-PROGRAM.
    MOVE ZERO TO ES-NULL-FLAG-P1.
READ-TEMPPI
    CALL "INPFIL" USING FCB-TEMP-FILE,
        RET-STATUS,
        WS-ES-REC,
        WS-ES-BUFFER-LENGTH,
        RETURN-LENGTH.
    IF RET-STATUS = KES-END-OF-FILE-INPUT
        PERFORM STARTPI-ZCHK
        COMPUTE ES-VAR-P2-P1 = WS-SUM / WS-COUNT
        MOVE ZERO TO WS-SUM, WS-COUNT
        GO TO READ-TEMPPI-EXIT.
    IF RET-STATUS NOT = KES-FILE-OK
        MOVE "ERROR OPENING FILE CDMTMPFL" TO MESG-DESC
        PERFORM PROCESS-ERROR
        GO TO EXIT-PROGRAM.
    IF WS-NULL-FLAG-P1 NOT = 1
        ADD WS-VAR-P2-P1 TO WS-SUM
        ADD 1 TO WS-COUNT.
    GO TO READ-TEMPPI.
STARTPI-ZCHK
    IF WS-COUNT = ZERO
        MOVE 1 TO WS-COUNT
        MOVE 1 TO ES-NULL-FLAG-P1.
READ-TEMPPI-EXIT.
EXIT.
CONTPIA
    MOVE "K" TO DISPOSITION.
    CALL "CLSFIL" USING FCB-TEMP-FILE,
RET-STATUS,
DISPOSITION.
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR CLOSING FILE CDMTMPFL" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
CALL "CLSFIL" USING FCB-ES-TEMP,
RET-STATUS,
DISPOSITION.
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR CLOSING FILE CDMESRES" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.

*******************************************************************************
LIBRARY NAME: VAX
MACRO NAME: CTOE18
PARAMETER:

    NEXT SENTENCE
    ELSE
    GO TO CS-ES-RTN.
LIBRARY NAME: VAX
MACRO NAME: CTOE19

PARAMETER:
  MOVE WS-ES-REC TO ES-TEMP-REC.
  CALL "OUTFIL" USING FCB-ES-TEMP,
       RET-STATUS,
       ES-TEMP-REC,
       ES-RECORD-LENGTH.
  IF RET-STATUS NOT = KES-FILE-OK
     MOVE "ERROR WRITING TO FILE CDMESRES"
        TO MESG-DESC
     PERFORM PROCESS-ERROR
     GO TO EXIT-PROGRAM.
  GO TO RELEASE-RECORDS.
LIBRARY NAME: VAX
MACRO NAME: CTOE20
PARAMETER:
    IF WS-ES-REC = DST-REC
       GO TO RELEASE-RECORDS.
    MOVE WS-ES-REC TO DST-REC.
SECTION 19

Function PRE8C - Generate CS-Selector Program

This function generates COBOL source code according to the ANSI X3.23-1974 standard which, at runtime performs the final qualification on conceptual rows, a file at a time, for the inner SELECT statements of a compound SELECT statement. There are no CS-ES transforms performed by the CS-Selector.

19.1 Inputs

1. TARGET-HOST PIC XXX
   Host upon which the CS-Selector program will execute at runtime.

2. MY-HOST PIC XXX
   Host upon which CDPRE8C executes at precompile time.

3. MOD-NAME PIC X(10)
   The program identification name of the CS-Selector program.

4. CS-ACTION-LIST included in CSAL copy member
   Conceptual representation of the fields to be retrieved.

5. CS-QUALIFY-LIST included in CSQUAL copy member
   Conceptual representation of the WHERE clause.

6. BOOLEAN-LIST
   Contains information about boolean operators and parenthesized logic from the WHERE clause.

7. IS-QUALIFY-LIST
   Internal representation of the WHERE clause.

8. ES-ACTION-LIST.
   External representation of the fields to be retrieved.
19.2 CDM Requirements

None

19.3 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 to be generated. This file must be closed prior to the CDMACR call. Upon return to CDPRE8C, 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 CDPRE8C input parameter TARGET-HOST.

MACRO-NAME contains the name of the macro to be generated, for example CSSEL01.

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 by setting 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.

19.4 Processing

1. Generate a unique file name to contain the generated 
COBOL code by calling GENFIL. GENFIL requires 
MY-HOST as an input parameter and returns the 30 
character file name and the 5 character status. 
This file name must be moved to the CDPRE8C output 
parameter GEN-FILE-NAM.

2. Determine which case is being handled. The case 
definitions are:

CASE 1 - A conceptual IF must be generated for final 
qualification.

CASE 1 applies when at least 1 used IS-QUALIFY 
entry has ISQ-EVAL-FLAG equal zero.

CASE 2 - No conceptual IF is to be generated.

CASE 2 applies when no used ISQ-EVAL-FLAG has a 
zero value.

CASE 3 - Code to distinct the results must be generated.

CASE 3 applies when there was a DISTINCT on an 
inner select of a combination query, or if 
distinct rows were specified to be selected when 
the external view was created. In either of 
these cases, ES-DISTINCT-FLAG will be set to 
"Y".

3. Processing for CASE 1

3.1 Generate the Identification Division through part of 
the file section by substituting the contents of 
CDPRE8C input parameter MOD-NAME for parameter P1 in 
macro 
CSSEL01.

3.2 For each CS field, generate the CS null flags 
according to the following format:
05 CS-NUL-L-FLA-G-xx PIC 9.

where xx through yy are the values of CS-INDEX. The 05 must start in column 16.

3.3 Generate each CS field description using the CS-TYPE, CS-SIZE and CS-ND fields. Use routine CDPIC to generate the picture clauses.

03 CS-VARxx pic clause.

where xx through yy are the values of CS-INDEX and the pic clause is the picture clause generated by CDPIC.

3.4 Compute the conceptual schema record size by summing all used CS-SIZEs together. For each conceptual field, add 1 additional position of the field's null flag.

3.5 Generate the end of the file section through part of the linkage section by substituting the value computed in the previous step for parameter P1, the value contained in input parameter TARGET-HOST for P2 and the value contained in input parameter MOD-NAME for P3 in macro CSSEL02.

3.6 Generate the names and picture clauses for the Conceptual Schema qualify variables which will be passed to the generated program at runtime.

For each CSQ element with CSQ-AUCR equal zero, generate the following:

03 CSQ-VAR-nn pic clause.

where nn is the CSQ-INDEX value. Call CDPIC using the corresponding CSQ-L-TYPE, CSQ-L-SIZE and CSQ-L-ND to generate the picture clause.

3.7 Generate the beginning of the Procedure Division
using macro CSSEL03 which has no parameters.

3.8 Call CDGENIF to generate the IF clauses to perform the qualification on the returned conceptual rows. CDGENIF requires the following parameters:

Input
- BOOLEAN-LIST
- CS-QUALIFY-LIST
- CS-ACTION-LIST
- IS-QUALIFY-LIST
- FILE-NAME

FILE-NAME must contain the file name generated in step 1. This file must be closed prior to the CDGENIF call.

If CDGENIF is successful (RET-STATUS equals KES-SUCCESSFUL) generate on the reopened for EXTEND file, the macro CSSEL04 which terminates the program.

Processing is now complete for CASE 1.

4. Processing for CASE 2

Generate the complete CASE 2 CS-Selector program by substituting the value of CDPRE8C input parameter MOD-NAME for parameter P1 and the value contained in input parameter TARGET-HOST for P2 in macro CSSEL05.

Processing is now complete for CASE 2.

5. Processing for CASE 3

5.1 Calculate the conceptual schema record size by summing all used CS-SIZEs together. For each conceptual field, add 1 additional position of the fields null flag.

5.2 Generate the Identification Division through part of the WORKING-STORAGE section by substituting the contents of CDPRE8C input parameter MOD-NAME for parameter P1 and the value calculated in the previous step for parameter P2 in macro CSSEL06.

5.3 For each CS field, generate the CS null flags according to the following format:
05 CS-NULL-FLAG-xx PIC 9.
   ...
   ...
   ...

05 CS-NULL-FLAG-yy PIC 9.

where xx through yy are the values of CS-INDEX. The 05 must start in column 16.

5.4 Generate each CS field description using the CS-TYPE, CS-SIZE and CS-ND fields. Use routine CDPIC to generate the picture clauses.

   03 CS-VARxx pic clause.
   ...
   ...
   ...

03 CS-VARyy pic clause.

where xx through yy are the values of CS-INDEX and pic clause is the picture clause generated by CDPIC.

5.5 A sort buffer must be generated. To generate the first part of the sort buffer, use macro CTOE4B (this macro is shared with CDPRE8) substituting the value 1 for P1 and 2 times the number of non-deleted CS entries for P2.

5.6 For each CS field, generate 2 sort buffer elements using macro CTOE4C, one for the field's null flag with one for the CS field itself. (Macro CTOE4C is shared with CDPRE8.)

To generate the sort buffer for a field's null flag, use macro CTOE4C, substituting the value of CS-INDEX for P1 (sort key starting position), the value 1 for P2 (sort key length), the value N for P3 (sort key type) and the value A for P4 (ascending sort).

To generate the sort buffer for the field itself, a running total must be kept of CS-SIZEs. This value
will be used in the calculation of the field's starting position. In macro CT0E4C, add 1, CS-USED and the running total described above to generate the value to substitute for P1.

Substitute the value of the current CS-SIZE for P2 (sort key length), the value of the current CS-TYPE for P3 (sort key type) and the value A for P4 (ascending sort).

5.7 Generate the end of the file section through part of the linkage section by substituting the value computed in the previous step for parameter P1, the value contained in input parameter TARGET-HOST for P2 and the value contained in input parameter MOD-NAME for P3 in macro CSSEL02.

5.8 Generate the names and picture clauses for the Conceptual Schema qualify variables which will be passed to the generated program at runtime.

For each CSQ element with CSQ-AUCR equal zero, generate the following:

03 CSQ-VAR-nn pic clause.

where nn is the CSQ-INDEX value. Call CDPIC using the corresponding CSQ-L-TYPE, CSQ-L-SIZE and CSQ-L-ND to generate the picture clause.

If there are no CSQ elements with CSQ-AUCR equal zero, generate:

03 CSQ-VAR-01 PIC X.

5.9 Generate the beginning of the Procedure Division, two calls to NAMFIL, two calls to OPNFIL, and one call to INPFIL using macro CSSEL07 which has no parameters.

5.10 If at least 1 used IS-QUALIFY entry has ISQ-EVAL-FLAG equal zero, perform the following two steps:

5.10.1 Call CDGENIF to generate the IF clauses to perform the qualification on the returned conceptual rows. CDGENIF requires the following parameters:
Input
- BOOLEAN-LIST
- CS-QUALIFY-LIST
- CS-ACTION-LIST
- IS-QUALIFY-LIST
- FILE-NAME PIC X(30).

5.10.2 FILE-NAME must contain the file name generated in step 1. This file must be closed prior to the CDGENIF call.

If CDGENIF is successful (RET-STATUS equals KES-SUCCESSFUL) generate on the reopened for EXTEND files the macro CSSEL08.

5.11 Generate a call to OUTFIL, the call to CDMPSOR, and the logic to transfer distinct records from the temporary file to the output file using macro CSSEL09 which has no parameters.

19.5 Outputs

1. GEN-FILE-NAME PIC X(30)

   The file name containing the generated COBOL program.

2. RET-STATUS PIC X(5)

   Error Status. A value equal to KES-SUCCESSFUL as defined in the ERRCDM copy member indicates successful completion.
Macro - CSSEL01
Library Name - VAX
Parameters - P1

IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
ENVIRONMENT DIVISION.
DATA DIVISION.

* WORKING-STORAGE SECTION.
  01 CS-INREC.
    03 CS-IN-NULL-FLAGS.
Macro - CSSEL02

Library Name - VAX

Parameters - P1, P2, P3

01 CS-OUTREC PIC X(P1).
   * 01 CDMCSRES PIC X(80).
   01 CDMCSOUT PIC X(80).
   01 MY-HOST PIC XXX VALUE "P2".
   01 MESG-DESC PIC X(60) VALUE SPACES.
   01 MODULE-NAME PIC X(10) VALUE "P3".
   01 FCB-CS-INPUT PIC S9(9) COMP.
   01 FCB-CS-OUTPUT PIC S9(9) COMP.
   01 DISPOSITION PIC X.
   01 NUMBER-OF-RECORDS PIC S9(9) COMP VALUE 2000.
   01 CS-RECORD-LENGTH PIC X9(9) COMP.
   01 CS-RETURN-LENGTH PIC S9(9) COMP.
   01 CS-OUT-RECORD-LENGTH PIC S9(9) COMP VALUE P1.

COPY CHKCDM OF IISSCLIB.
COPY ERRCDM OF IISSCLIB.
COPY ERRFS OF IISSCLIB.

* LINKAGE SECTION.
   01 IN-FILE-NAME PIC X(80).
   01 IN-COUNT PIC S9(9) COMP.
   01 OUT-FILE-NAME PIC X(80).
   01 OUT-COUNT PIC S9(9) COMP.
   01 RET-STATUS PIC X(5).
   01 CS-QUALIFY-VAR.
Macro - CSSEL03

Library Name - VAX

Parameters - none

PROCEDURE DIVISION USING IN-FILE-NAME,
IN-COUNT,
CS-QUALIFY-VAR,
OUT-FILE-NAME,
OUT-COUNT,
RET-STATUS.

START-PROGRAM.
MOVE ZERO TO OUT-COUNT.
MOVE KES-SUCCESSFUL TO RET-STATUS.
MOVE IN-FILE-NAME TO CDMCSRES.
CALL "NAMFIL" USING OUT-FILE-NAME.
IF OUT-FILE-NAME EQUAL LOW-VALUE
   MOVE "UNABLE TO GENERATE OUTFILE" TO MESG-DESC
   PERFORM PROCESS-ERROR
   GO TO EXIT-PROGRAM.
MOVE OUT-FILE-NAME TO CDMCSOUT.
MOVE "R" TO DISPOSITION.
CALL "OPNFIL" USING FCB-CS-INPUT,
   RET-STATUS,
   CDMCSRES,
   DISPOSITION,
   CS-RECORD-LENGTH,
   NUMBER-OF-RECORDS.
IF RET-STATUS NOT = KES-FILE-OK
   MOVE "ERROR OPENING CDMCSRES" TO MESG-DESC
   PERFORM PROCESS-ERROR
   GO TO EXIT-PROGRAM.
MOVE "W" TO DISPOSITION.
CALL "OPNFIL" USING FCB-CS-OUTPUT,
   RET-STATUS,
   CDMCSOUT,
   DISPOSITION,
   CS-OUT-RECORD-LENGTH,
   NUMBER-OF-RECORDS.
IF RET-STATUS NOT = KES-FILE-OK
   MOVE "ERROR OPENING CDMCSOUT" TO MESG-DESC
   PERFORM PROCESS-ERROR
   GO TO EXIT-PROGRAM.
CS-SEL-RTN.
CALL "INPFIL" USING FCB-CS-INPUT,
RET-STATUS,
CS-INREC,
CS-RECORD-LENGTH,
CS-RETURN-LENGTH.

IF RET-STATUS = KES-END-OF-FILE-INPUT
GO TO EXIT-PROGRAM.

IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR READING CDMCSRES" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
NEXT SENTENCE
ELSE
   GO TO CS-SEL-RTN.
   MOVE CS-INREC TO CS-OUTREC.
   CALL "OUTFIL" USING FCB-CS-OUTPUT,
       RET-STATUS,
       CS-OUTREC,
       CS-OUT-RECORD-LENGTH.
   IF RET-STATUS NOT = KES-FILE-OK
      STRING "CS-OUTREC WRITE ERROR: " RET-STATUS
      DELIMITED BY SIZE INTO MESG-DESC
      PERFORM PROCESS-ERROR
      GO TO REAL-EXIT-PROGRAM.
   ADD 1 TO OUT-COUNT.
   GO TO CS-SEL-RTN.
EXIT-PROGRAM.
   MOVE "K" TO DISPOSITION.
   CALL "CLSFIL" USING FCB-CS-INPUT,
       RET-STATUS,
       DISPOSITION.
   IF RET-STATUS NOT = KES-FILE-OK
      STRING "RESULTS FILE CLOSE ERROR: " RET-STATUS
      DELIMITED BY SIZE INTO MESG-DESC
      PERFORM PROCESS-ERROR
      GO TO REAL-EXIT-PROGRAM.
   CALL "CLSFIL" USING FCB-CS-OUTPUT,
       RET-STATUS,
       DISPOSITION.
   IF RET-STATUS NOT = KES-FILE-OK
      STRING "RESULTS FILE CLOSE ERROR: " RET-STATUS
      DELIMITED BY SIZE INTO MESG-DESC
      PERFORM PROCESS-ERROR
      GO TO REAL-EXIT-PROGRAM.
   CALL "DELFIL" USING MY-HOST CDMCSRES.
REAL-EXIT-PROGRAM.
EXIT PROGRAM.
COPY ERRPRO OF IISSCLIB.
Macro Name - CSSEL05
Library Name - VAX
Parameters - P1
     P2

CS SELECTOR CODE - CASE 2 (NO IF)

IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
  01 MY-HOST PIC XXX VALUE "P2".
  01 MESS-DESC PIC X(60) VALUE SPACES.
  01 MODULE-NAME PIC X(10) VALUE "P1".
COPY CHKCDM OF IISSCLIB.
COPY ERRCDM OF IISSCLIB.
COPY ERRFS OF IISSCLIB.

LINKAGE SECTION.
  01 IN-FILE-NAME PIC X(80).
  01 IN-COUNT PIC S9(9) COMP.
  01 OUT-FILE-NAME PIC X(80).
  01 OUT-COUNT PIC S9(9) COMP.
  01 RET-STATUS PIC X(5).
     03 FILLER PIC X.

PROCEDURE DIVISION USING IN-FILE-NAME,
     IN-COUNT,
     CS-QUALIFY-VAR,
     OUT-FILE-NAME,
     OUT-COUNT,
     RET-STATUS.
START-PROGRAM.
    MOVE IN-FILE-NAME TO OUT-FILE-NAME.
    MOVE IN-COUNT TO OUT-COUNT.
    MOVE KES-SUCCESSFUL TO RET-STATUS.
EXIT-PROGRAM.
    EXIT PROGRAM.
COPY ERRPRO OF IISSCLIB.
MACRO NAME - CSSEL06

LIBRARY NAME - VAX

PARAMETERS - P1  P2

IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
* This program distincts retrieved conceptual data.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 TEMP-REC            PIC X(P2).
01 DST-REC             PIC X(P2).
01 CDMTMPF1            PIC X(80) VALUE SPACES.
01 CDMTMPF2            PIC X(80) VALUE SPACES.
01 FCB-TEMP-1          PIC S9(9) COMP.
01 FCB-TEMP-2          PIC S9(9) COMP.
01 TEMP-RECORD-LENGTH  PIC S9(9) COMP.
* 01 CS-REC.           03 CS-NUL-FLAGS.

**************************
MACRO NAME - CSSEL07

LIBRARY NAME - VAX

PARAMETERS - NONE

*BEGINNING OF PROCEDURE DIVISION FOR CASE 3 WHERE
*THE DISTINCT FLAG IS SET.

PROCEDURE DIVISION USING IN-FILE-NAME,
    IN-COUNT,
    CS-QUALIFY-VAR,
    OUT-FILE-NAME,
    OUT-COUNT,
    RET-STATUS.

START-PROGRAM.

MOVE SPACES TO CS-OUTREC.
MOVE ZERO TO OUT-COUNT.
MOVE KE$-SUCCESSFUL TO RET-STATUS.
MOVE IN-FILE-NAME TO CDMCSRES.
CALL "NAMFIL" USING CDMCSOUT.
IF CDMCSOUT = LOW-VALUE
    MOVE "UNABLE TO GENERATE OUTFILE"
    TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
CALL "NAMFIL" USING CDMTMPF1.
IF CDMTMPF1 = LOW-VALUE
    MOVE "UNABLE TO GENERATE TEMPFILE1"
    TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
CALL "NAMFIL" USING CDMTMPF2.
IF CDMTMPF2 = LOW-VALUE
    MOVE "UNABLE TO GENERATE TEMPFILE2"
    TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.

MOVE "R" TO DISPOSITION.
CALL "OPNFIL" USING FCB-CS-INPUT,
    RET-STATUS,
    CDMCSRES,
    DISPOSITION,
    CS-RECORD-LENGTH,
    NUMBER-OF-RECORDS.
IF RET-STATUS NOT = KES-FILE-OK
  MOVE "ERROR OPENING FILE CDMCSRES" TO MESG-DESC
  PERFORM PROCESS-ERROR
  GO TO EXIT-PROGRAM.
MOVE "W" TO DISPOSITION.
CALL "OPNFIL" USING FCB-TEMP-1,
  RET-STATUS,
  CDMTMPF1,
  DISPOSITION,
  CS-RECORD-LENGTH,
  NUMBER-OF-RECORDS.
IF RET-STATUS NOT = KES-FILE-OK
  MOVE "ERROR OPENING FILE CDMTMPF1" TO MESG-DESC
  PERFORM PROCESS-ERROR
  GO TO EXIT-PROGRAM.
CS-SEL-RTN.
CALL "INPFIL" USING FCB-CS-INPUT,
  RET-STATUS,
  CS-REC,
  CS-RECORD-LENGTH,
  CS-RETURN-LENGTH.
IF RET-STATUS = KES-END-OF-FILE-INPUT
  MOVE "K" TO DISPOSITION
  CALL "CLSFIL" USING FCB-CS-INPUT,
    RET-STATUS,
    DISPOSITION
IF RET-STATUS NOT = KES-FILE-OK
  MOVE "ERROR CLOSING FILE CDMCSRES" TO MESG-DESC
  PERFORM PROCESS-ERROR
  GO TO EXIT-PROGRAM
ELSE
  CALL "CLSFIL" USING FCB-TEMP-1,
    RET-STATUS,
    DISPOSITION
IF RET-STATUS NOT = KES-FILE-OK
  MOVE "ERROR CLOSING FILE CDMTMPF1" TO MESG-DESC
  PERFORM PROCESS-ERROR
  GO TO EXIT-PROGRAM
ELSE
  GO TO CS-SEL-RTN-EXIT.
IF RET-STATUS NOT = KES-FILE-OK
  MOVE "ERROR READING FILE CDMCSREC" TO MESG-DESC
  PERFORM PROCESS-ERROR
  GO TO EXIT-PROGRAM.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
MACRO NAME - CSSEL08
LIBRARY NAME - VAX
PARAMETERS - NONE
    NEXT SENTENCE
ELSE
    GO TO CS-SEL-RTN.
MACRO NAME - CSSEL09
LIBRARY NAME - VAX
PARAMETERS - NONE

MOVE CS-REC TO TEMP-REC.
CALL "OUTFIL" USING FCB-TEMP-1,
    RET-STATUS,
    TEMP-REC,
    CS-RECORD-LENGTH.

IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR WRITING TO FILE CDMTMPF1" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
GO TO CS-SEL-RTN.
CS-SEL-RTN-EXIT.
EXIT.
START-SORT.
MOVE CDMTMPF1 TO FILE-NAME-I.
CALL "CDMPSOR" USING INPUT-FILE-i,
    CDMTMPF2,
    *    MESG-DESC,
    RET-STATUS.
MOVE RET-STATUS TO QCS-CDMP-CHECK-STATUS.
IF NOT QCS-SUCCESSFUL
    MOVE "SORT/MERGE PROGRAM FAILED" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
MOVE "R" TO DISPOSITION.
CALL "OPNFIL" USING FCB-TEMP-2,
    RET-STATUS,
    CDMTMPF2,
    DISPOSITION,
    TEMP-RECORD-LENGTH,
    NUMBER-OF-RECORDS.

IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR OPENING FILE CDMTMPF2" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
MOVE SPACES TO DST-REC.
RELEASE-RECORDS.
MOVE SPACES TO TEMP-REC.
CALL "INPFIL" USING FCB-TEMP-2,
    RET-STATUS,
    TEMP-REC,
    TEMP-RECORD-LENGTH,
    CS-RETURN-LENGTH.
IF RET-STATUS = KES-END-OF-FILE-INPUT
MOVE "K" TO DISPOSITION
CALL "CLSFIL" USNG FCB-TEMP-2,
   RET-STATUS,
   DISPOSITION
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR CLOSING FILE CDMTMPF2" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM
ELSE
   CALL "CLSFIL" USING FCB-CS-OUTPUT,
   RET-STATUS,
   DISPOSITION
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR CLOSING FILE CDMCSOUT" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM
ELSE
   GO TO EXIT-PROGRAM.
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR READING FILE CDMTMPF2" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
************************************************************
IF TEMP-REC = DST-REC
GO TO RELEASE-RECORDS.
ADD 1 TO OUT-COUNT.
MOVE TEMP-REC TO DST-REC.
MOVE TEMP-REC TO CS-OUTREC.
CALL "OUTFIL" USING FCB-CS-OUTPUT,
   RET-STATUS,
   CS-OUTREC,
   TEMP-RECORD-LENGTH.
IF RET-STATUS NOT = KES-FILE-OK
MOVE "ERROR WRITING TO FILE CDMCSOUT" TO MESG-DESC
PERFORM PROCESS-ERROR
GO TO EXIT-PROGRAM.
GO TO RELEASE-RECORDS.
EXIT-PROGRAM.
   MOVE SPACES TO CS-REC, CS-OUTREC, TEMP-REC.
PEFORM DEL-PARA.
END-PROGRAM.
EXIT PROGRAM.
COPY ERRPRO OF IISSCLIB.
DEL-PARA.
   CALL "DELFIL" USING MY-HOST,
       CDMCSRES.
   CALL "DELFIL" USING MY-HOST,
CALL "DELFIL" USING MY-HOST,
CDMTMPF2.
SECTION 20

Function PRE8D - Generate Referential Integrity Test and Key Uniqueness Program

This function generates COBOL source code according to the ANSI X3.23-1974 standard, which at runtime performs the final qualification on type 1 and type 2 referential integrity tests and key uniqueness tests.

20.1 Inputs

1. TARGET-HOST PIC XXX
   Host upon which the Type 2 R.I. Program will execute at runtime.

2. MY-HOST PIC XXX
   Host upon which CDPRE8D executes at precompile time.

3. MOD-NAME PIC X(10)
   The program identification name of the Type 2 R.I. Program.

4. CS-ACTION-LIST included in CSAL copy member
   Conceptual representation of fields to be deleted.

5. CS-QUALIFY-LIST included in CSQUAL copy member
   Conceptual representation of the WHERE clause.

6. BOOLEAN-LIST
   Contains information about boolean operators and parenthesized logic from the WHERE clause.

7. IS-QUALIFY-LIST
   Internal representation of the WHERE clause.

20.2 CDM Requirements

None

20.3 Internal Requirements

20-1
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 to be generated. This file must be closed prior to the CDMACR call. Upon return to CDPRE8D, **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 CDPRE8D input parameter **TARGET-HOST**.

**MACRO-NAME** contains the name of the macro to be generated, for example T2RI01.

**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, for example P1. **SL-SUBST-VAL** (index) contains the corresponding substitution value, for example CS-NDML-NO.
20.4 Processing

1. Generate a unique file name to contain the generated COBOL code by calling GENFIL. GENFIL requires MY-HOST as an input parameter and returns the 30 character file name and the 5 character status. This file name must be moved to the CDPRE8D output parameter GEN-FILE-NAME.

2. Determine which case is being handled. The case definitions are:

CASE 1 - A conceptual IF must be generated for final qualification.

CASE 1 applies when at least 1 used IS-QUALIFY entry has ISQ-EVAL-FLAG equal zero.

CASE 2 - No conceptual IF is to be generated.

CASE 2 applies when no used ISQ-EVAL-FLAG has a zero value.

3. Processing For CASE 1

3.1 Generate the Identification Division through part of the file section by substituting the contents of CDPRE8D input parameter MOD-NAME for parameter P1 in macro T2RI01.

3.2 For each CS field, generate the CS null flags according to the following format:

\[
05 \text{ CS-NULL-FLAG-xx } \text{ PIC 9.} \\
\ldots \\
\ldots \\
05 \text{ CS-NULL-FLAG-yy } \text{ PIC 9.}
\]

where xx through yy are the values of CS-INDEX. The 05 must start in column 16.

3.3 Generate each CS field description using the CS-TYPE, CS-SIZE and CS-ND fields. Use routine CDPIC to generate the picture clauses.

\[
03 \text{ CS-VARxx } \text{ pic clause.}
\]
03 CS-VARyy pic clause.

where xx through yy are the values of CS-INDEX and pic clause is the picture clause generated by CDP1C.

3.4 Generate the working storage section through part of the linkage section by substituting the value of CDPRE88D input parameter TARGET-HOST for P1 and the value of input parameter MOD-NAME for P2 in macro T2RI02.

3.5 Generate the names and picture clauses for the conceptual schema qualify variables which will be passed to the generated program at runtime.

Scan the CS-QUALIFY-LIST searching for a zero value in a used CSQ-AUCR. For each CSQ element with CSQ-AUCR equal zero, generate the following:

03 CSQ-VAR-nn pic clause.

where nn is the CSQ-INDEX value. Call CDPIC using the corresponding CSQ-L-TYPE, CSQ-L-SIZE and CSQ-L-ND to generate the picture clause.

3.6 Generate the beginning of the Procedure Division using macro T2RI03 which has no parameters.

3.7 Call CDGENIF to generate the IF clauses to perform the final qualification on the returned conceptual rows. CDGENIF requires the following parameters:

Input
   BOOLEAN-LIST
   CS-QUALIFY-LIST
   DUMMY               PIC X
   QUALIFY-TYPE        PIC X VALUE "C"
   FILE-NAME           PIC X(30)
   SUBTRANS-ID         PIC 999 VALUE ZERO
   DUMMY               PIC X

Output
   RET-STATUS          PIC X(5)

FILE-NAME must contain the file name generated in step 1. This file must be closed prior to

20-4
the CDGENIF call.

3.8 Generate on the reopened for EXTEND file, the macro T2RI04 which has no parameters and which terminates the generated program.

Processing for CASE 1 is complete.

4. Processing For CASE 2

Generate the complete CASE 2 Type 2 referential integrity checker by substituting the value of CDPRE8D input parameter MOD-NAME for parameter P1 and the value of input parameter TARGET-HOST for P2 in macro T2RI05.

Processing is complete for CASE 2.

20.5 Outputs

1. GEN-FILE-NAME PIC X(30)

   The file name containing the generated COBOL Type 2 R.I. Program.

2. RET-STATUS PIC X(5)

   Error Status. A value equal to KES-SUCCESSFUL as defined in the ERRCDM copy member indicates successful completion.
Macro T2RI01

Library Name - VAX

Parameters - P1

IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION
01 CS-REC.
   03 CS-NUL-FLAGS.
**Macro T2RI02**

**Library Name** - VAX

**Parameters** - P1, P2

```asm
01 CDMCSRES PIC X(80).
01 MY-HOST PIC X(60) VALUE "P1".
01 MESG-DESC PIC X(80) VALUE SPACES.
01 MODULE-NAME PIC X(10) VALUE "P2".
01 DISPOSITION PIC X.
01 FCB-CS-INPUT PIC S9(9) COMP.
01 CS-RECORD-LENGTH PIC S9(9) COMP.
01 NUMBER-OF-RECORDS PIC S9(9) COMP VALUE 2000.
01 RETURN-LENGTH PIC S9(9) COMP.
COPY CHKCDM OF IISSCLIB.
COPY ERRCDM OF IISSCLIB.
COPY ERRFS OF IISSCLIB.

**LINKAGE SECTION.**
01 CDM-CS-RESULTS-FILE PIC X(80).
01 RI-COUNT PIC 9(6).
01 RET-STATUS PIC X(5).
01 CS-QUALIFY-VAR.
```
Macro T2RI03

Library Name - VAX

Parameters - none

PROCEDURE DIVISION USING CDM-CS-RESULTS-FILE,
   CS-QUALIFY-VAR,
   *
   RI-COUNT,
   RET-STATUS.

START PROGRAM.
   MOVE ZERO TO RI-COUNT.
   MOVE KES-SUCCESSFUL TO RET-STATUS.
   MOVE CDM-CS-RESULTS-FILE TO "CDMCSRES".
   MOVE "R" TO DISPOSITION.
   CALL "OPNFIL" USING FCB-CS-INPUT,
      RET-STATUS,
      CDMCSRES,
      DISPOSITION,
      CS-RECORD-LENGTH,
      NUMBER-OF-RECORDS.

IF RET-STATUS NOT = KES-FILE-OK
   MOVE "ERROR OPENING FILE CDMCSRES" TO MESG-DESC
   PERFORM PROCESS-ERROR
   GO TO EXIT-PROGRAM.
CS-RI2-RTN.
   CALL "INPFIL" USING FCB-CS-INPUT,
      RET-STATUS,
      CS-REC,
      CS-RECORD-LENGTH,
      RETURN-LENGTH.

IF RET-STATUS = KES-END-OF-FILE-INPUT,
   GO TO EXIT-PROGRAM.
IF RET-STATUS NOT = KES-FILE-OK
   MOVE "ERROR READING FILE CDMCSRES" TO MESG-DESC
   PERFORM PROCESS-ERROR
   GO TO EXIT-PROGRAM.
Macro T2RI04

Library Name - VAX

Parameters - none

```
MOVE 1 TO RI-COUNT
GO TO EXIT-PROGRAM
ELSE
   GO TO CS-RI2-RTN.
EXIT-PROGRAM.
   MOVE "K" TO DISPOSITION.
   CALL "CLSFIL" USING FCB-CS-INPUT,
      RET-STATUS,
      DISPOSITION.
   IF RET-STATUS NOT = KES-FILE-OK
      MOVE "EROR CLOSING FILE CDMCSRES" TO MESG-DESC
      PERFORM ERROR-PROCESS
   ELSE
      CALL "DELFIL" USING MY-HOST, CDMCSRES.
      EXIT PROGRAM.
COPY ERRPRO OF IISSCLIB.
```
Macro T2RI05

Library Name - VAX

Parameters - P1
P2

IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 MY-HOST PIC XXX VALUE "P2".
01 MESG-DESC PIC X(60) VALUE SPACES.
01 MODULE-NAME PIC X(10) VALUE "P1".
COPY CHKCDM OF IISSCLIB.
COPY ERRCDM OF IISSCLIB.
COPY ERRFS OF IISSCLIB.

LINKAGE SECTION.
01 CDM-CS-RESULTS-FILE PIC X(80).
01 RI-COUNT PIC 9(6).
01 RET-STATUS PIC X(5).
01 CS-QUALIFY-VAR.
03 FILLER PIC X.

PROCEDURE DIVISION USING CDM-CS-RESULTS-FILE,
CS-QUALIFY-VAR,
*  
   RI-COUNT,
   RET-STATUS.

START PROGRAM.
   MOVE 1 TO RI-COUNT.
   MOVE KES-SUCCESSFUL TO RET-STATUS.

EXIT-PROGRAM.
   CALL "DELFIL" USING MY-HOST, CDM-CS-RESULTS-FILE.
   EXIT PROGRAM.

COPY ERRPRO OF IISSCLIB.