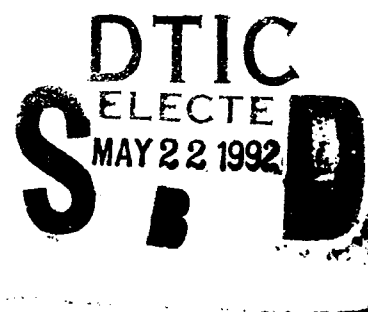# AD-A250 463

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 26 - Distributed Request Supervisor Development Specification

J. Althoff, M. Apicella

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

DTIC
ELECTE
MAY 22 1992
S   B   D

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533
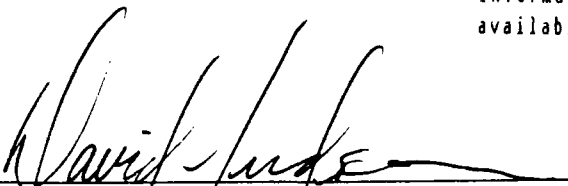
92-13581

92 5 21

NOTICE

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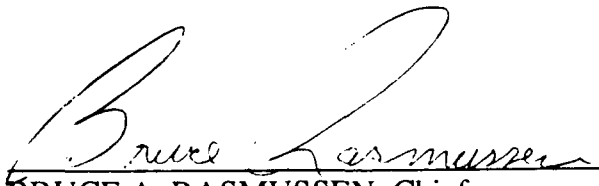
This technical report has been reviewed and is approved for publication.

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

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

DATE 25 July 91

FOR THE COMMANDER:

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

DATE 25 July 91

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

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

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS		

2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT
		
		    Approved for Public Release;
		    Distribution is Unlimited.		

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE				

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
    DS 620341310
		5. MONITORING ORGANIZATION REPORT NUMBER(S)
		    WRDC-TR-90-8007 Vol. V, Part 26		

6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services	6b. OFFICE SYMBOL *(if applicable)*	7a. NAME OF MONITORING ORGANIZATION WRDC/MTI		

6c. ADDRESS *(City, State, and ZIP Code)*
    2970 Presidential Drive
    Fairborn, OH 45324-6209
		7b. ADDRESS *(City, State, and ZIP Code)*
		
		    WPAFB, OH 45433-6533		

8a. NAME OF FUNDING/SPONSORING
    ORGANIZATION
Wright Research and Development Center,
Air Force Systems Command, USAF
	8b. OFFICE SYMBOL
	(if applicable)
	
	WRDC/MTI
		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM.
		
		    F33600-87-C-0464		

8c. ADDRESS *(City, State, and ZIP Code)* Wright-Patterson AFB, Ohio 45433	10. SOURCE OF FUNDING NOS.			
11. TITLE *(Include Security Classification)* See block 19	PROGRAM ELEMENT NO. 78011F	PROJECT NO. 595600	TASK NO. F95600	WORK UNIT NO. 20950607

12. PERSONAL AUTHOR(S)
    Control Data Corporation: Althoff, J. L., Apicella, M. L.

13a. TYPE OF REPORT Final Report	13b. TIME COVERED 4/1/87-12/31/90	14 DATE OF REPORT *(Yr.,Mo.,Day)* 1990 September 30	15. PAGE COUNT 32

16. SUPPLEMENTARY NOTATION

    WRDC/MTI Project Priority 6203

17.	COSATI CODES		18. SUBJECT TERMS *(Continue on reverse if necessary and identify block no.)*
FIELD	GROUP	SUB GR.	
1308	0905		

19. ABSTRACT *(Continue on reverse if necessary and identify block number)*

This Development Specification (DS) describes the functions, performance, environment, interfaces, test, qualification, and design requirements for the Distributed Request Supervisor computer program.

    BLOCK 11:

    INTEGRATED INFORMATION SUPPORT SYSTEM
    Vol V - Common Data Model Subsystem

    Part 26 - Distributed Request Supervisor Development
            Specification

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson	22b. TELEPHONE NO. *(Include Area Code)* (513) 255-7371	22c. OFFICE SYMBOL WRDC/MTI

FOREWORD

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

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

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

SUBCONTRACTOR	ROLE
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface,and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

Table of Contents

SECTION 1

SCOPE

1.1 Identification

This specification establishes the performance, develop-
ment, test, and qualification requirements of a collection of
computer programs identified as Configuration Item "Distributed
Request Supervisor."

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

Common Data Model Processor (CDMP)

The CDMP consists of three CIs that manage users' accesses
to distributed databases in IISS. Input to the CDMP consists
of user transactions, which may be in the form of neutral data
manipulation language (NDML) commands embedded in COBOL host
programs or NDML commands phrased as stand-alone requests.
These development specifications address only the management of
embedded NDML commands.

The Precompiler CI parses the application program source
code, identifying NDML commands. It applies external-schema-
to-conceptual-schema transforms on the NDML command, and decom-
poses the conceptual schema command into single database re-
quests. These single database requests are each transformed
into programs (called Request Processors) to access the
specific databases to retrieve or update the data as required
by the NDML command. The NDML commands in the application
source program are replaced by function calls which, when
executed, will activate the run-time query evaluation processes
associated with the particular NDML command.

The Precompiler also generates a CS/ES Transformer program
which will take the final result of the query, stored in a file
as a conceptual schema relation, and transform it into the
appropriate external schema relation.

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

The Distributed Request Supervisor (DRS) CI is responsible for coordination of the run-time activity associated with the evaluation of an NDML command. It is activated by the application program, which sends it the names and locations of the Request Processors to be activated, along with run-time parameters which are to be sent to the Request Processors. The DRS activates the Request Processors, sending them the run-time parameters. The results of the Request Processor executions are stored as files, in the form of conceptual schema relations, on the hosts which executed the Request Processors. Using the Join Query Graph, transmission cost information and data about intermediate results, the DRS determines a good strategy for combining the intermediate results of the NDML command. It issues the appropriate file transfer requests, activates Aggregators to perform join, union, and outer join operations. Finally, the DRS notifies the application program that the query is completed, and sends it the name of the file which contains the results of the query.

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

The CDMP provides the application programmer with important capabilities to:

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

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

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

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

Please refer to the Software Availability Bulletin, Volume III, Part 16, CI# SAB620326000, for current IISS software and

documentation availability.

1.2 Functional Summary

The overall objectives of this CI are to:

a. Determine the appropriate sequence of inter-database JOIN, UNION and OUTER JOIN operations required to produce the result for a multi-database transaction.

b. Coordinate and control the interactions among a user's Application Process (AP), the generated Request Processors (RP) and the Aggregator(s) for both single- and multi-database transactions.

Determination of JOIN, UNION, OUTER JOIN

The DRS will calculate costs for each inter-site join, union, and outer join possibility, select the alternative with minimum cost and will generate the appropriate sequence of join, union, and outer join operations that will collapse the intermediate relations into the proper destination relation. The sequence is generated at run-time by the DRS at the node of the transaction's originating AP cluster.

The Distributed Request Supervisor receives information about a set of intermediate relations, which are the result of processing portions of a transaction at the local databases. Once all local processing is complete, the intermediate relations must be joined together. The Distributed Request Supervisor solves the problem of determining in which order these relations should be combined, which includes the sequencing of transmission of relations from one database to another in order to perform the operations in such a way as to minimize the amount of data transferred.   The operations are performed by the Aggregator CI.

In order to process a transaction efficiently, the Distributed Request Supervisor determines all possible inter-database operations and calculates the transmission costs for each possibility. It selects the operation with the least cost, sends the appropriate transmission commands to the identified Aggregator site, and updates information tables as each is performed. If a join or outer join is chosen as the next step to schedule, the two relations may not be partitions; only "whole" relations are joined. After the last operation is performed, the resulting relation is transmitted to the site at which the result is to appear. A CS/ES Transform process is then initialized to perform the required transformations.

AP/RP/Aggregator Coordination

Each user AP that contains NDML requests has a copy of the DRS, which it calls as a subroutine. All these copies are exactly the same. Each copy is responsible for the coordination

and control of all the local and remote RPs and Aggregators that are used to process the NDML requests from its user AP (see that it is called as a subroutine by the DRS and that it accesses a database on the same node as the user AP). A remote RP is one that is called via the NTM; the database it accesses may be on the same node as the user AP or on a different node. The DRS uses the NTM message handling facility to communicate with any remote RPs and Aggregators. It calls any local RPs and Aggregators via dynamic call capability of COBOL 74. This allows each copy of the DRS to (indirectly) call a variety of RPs, Aggregators, and CS-ES Transformers while still being identical to all other DRS copies (PRE15 assigns the same name to all the Subroutine Callers).

The initiation of a transaction occurs when the AP sends a message to the DRS to initiate activity on the specified transaction. The DRS will then initiate the proper DRS tables and Request Processors (RPs) in preparation for the first NDML request from the AP. When the AP makes an NDML request, the AP will 'go to sleep' and wait for the DRS response. The DRS will activate/reactivate the proper RP(s). The DRS will wait until the RPs have completed, then will decide if an Aggregator needs to be called. If so, the DRS will do so and wait until it receives a message from the Aggregator indicating completion. The DRS will then return directly to the AP.

Since the proper execution of recovery units in the local DBMSs requires that a specific AP communicate with a consistent instantiation of an RP during the life of the AP recovery unit, the DRS must guarantee that, once an AP initiates a recovery unit, the RP to which the first local DML request was sent (within the confines of the recovery unit) must be the RP to which all further local DML requests (for the specific DBMS/node) are sent, until the recovery unit termination request is received from the AP. Therefore, the DRS will maintain a table of the RPs for each AP with which the DRS is communicating, regardless of the type of NDML verbs being executed by the AP.

Compile-time Activities

Compile-time activities include building skeleton process information tables. At run-time, the Distributed Request Supervisor sends messages for the local Request Processor to begin processing. As the local Request Processor finishes its subtransactions, it sends back information for the Distributed Request Supervisor to use in deciding the sequence of steps to be taken. The Distributed Request Supervisor then, as needed, initiates file transfer requests and activates appropriate Aggregators.

The major functions to be described in this document for this CI are:

DRS1:   Initiate Subtransaction Processing

DRS2:   Schedule Stages

DRS3:   Transfer Final Results

SECTION 2

DOCUMENTS

2.1 Applicable Documents

Following is a list of applicable documents relating to this Computer Program Development Specification for the system identified as the Common Data Model Processor (CDMP) Distributed Request Supervisor.

Related ICAM Documents included:

UM620341001	CDM Administrator's Manual
TBM620341000	CDM1, An IDEF1 Model of the Common Data Model
UM620341100	Neutral Data Definition Language (NDDL) User's Guide
PRM620341200	Embedded NDML Programmer's Reference Manual
UM620341002	ICAM Definition Method for Data Modeling (IDEF1 - Extended)
DS620341200	Development Specification for the IISS NDML Precompiler Configuration Item
DS620341320	Development Specification for the IISS Aggregator Configuration Item

Other references include:

Astrahan, M.M. et al., "System R: Relational Approach to Database Management," ACM Transactions on Database Systems, Vol. 1, No. 2, June 1976, pp. 97-137.

Bernstein, P.A. and Chiu, D.M., "Using Semi-Joins to Solve Relational Queries," Journal for the Association for Computing Machinery, Vol. 28, No. 1, January 1981, pp. 25-40.

Bernstein, P.A. et al., "Query Processing in a System for Distributed Databases (SDD-1)," ACM Transactions on Database Systems, Vol. 6, No. 4, December 1981, pp. 602-625.

Chang, J.M. "A Heuristic Approach to Distributed Query Processing," Proceedings of the Eighth International

Conference on Very Large Data Bases, Mexico City, September 1982, pp. 54-61.

Epstein, R., M. Stonebraker, and E. Wong. "Distributed Query Processing in a Relational Database System," *Proceedings of the ACM SIGMOD International Conference*, Austin, June 1978, pp. 169-180.

Hevner, A.R. and S.B. Yao. "Query Processing in Distributed Database Systems," *IEEE Transactions on Software Engineering*, May 1979, pp. 177-187.

Rothnie, J.B. et al. "Introduction to a System for Distributed Databases," *ACM Transactions on Database Systems*, Vol. 5, No. 1, May 1980, pp. 1-17.

Rothnie, J.B. and N. Goodman. "A Survey of Research and Development in Distributed Database Management." *Proceedings Third International Conference on Very Large Databases*, Tokyo, 1977, pp. 48-62.

Takizawa. "Distributed Database System - JDDBS-1," *JIPDEC*, Japan, 1982.

Wong, E. and K. Youssefi. "Decomposition - A Strategy for Query Processing," *ACM Transactions on Database Systems*, Vol. 1, No. 3, September, 1976, pp. 223-241.

2.2 Terms and Abbreviations

The following acronyms are used in this document:

APL   Attribute Pair List

AUC   Attribute Use Class

CDMP Common Data Model Processor

CI   Configuration Item

CS   Conceptual Schema

DML   Data Manipulation Language

DRS   Distributed Request Supervisor
      (previously SS: Stager/Scheduler)

ES   External Schema

ICAM Integrated Computer Aided Manufacturing

IISS Integrated Information Support System

IS   Internal Schema

NDML Neutral Data Manipulation Language

RFT   Result Field Table

RP   Request Processor
      (previously QP: Query Processor)

SDS   System Design Specification

SECTION 3

REQUIREMENTS

3.1 Computer Program Definition

3.1.1 System Capacities

The DRS must operate within the capacity of the host computer and is functionally dependent upon NTM Services.

3.1.2 Interface Requirements

3.1.2.1 Interface Blocks

This CI is the mechanism that determines the order of aggregating intermediate results of accesses to distributed databases, and coordinates interactions among the AP/RP/DRS/Aggregators during updates.

There is a Distributed Request Supervisor program for each site or host in the IISS network. An instance of the Distributed Request Supervisor (DRS) program runs as a subroutine to each user AP that contains NDML commands. This DRS takes on the role of master control program for all the transactions from that user AP. Instances of Distributed Request Supervisor at other sites become the master control programs for transactions initiated at those sites.

The Distributed Request Supervisor CI has responsibility for run-time scheduling of activities comprising distributed database accesses and updates. It initates local Request Processors and receives replies when they have completed. It sends subtransactions to appropriate application clusters, initiates file transfer requests to transmit intermediate relations, and initiates corresponding Aggregators to perform join, union, and outer join operations on intermediate relations.

The interfaces of each Distributed Request Supervisor include input in the form of messages indicating completion of activities under the Distributed Request Supervisor control and the Join Query Graph from the Precompiler CI. Outputs are in the form of "staging sequences" that direct activities of other run-time modules.

3.1.2.2 Detail Interface Definition

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

The DRS depends heavily upon three capabilities of the

NTM, and without these facilities, will not properly function.

a. If a process dies or is killed, the NTM must notify, via an unsolicited message, the parent process.

b. If a process dies or is killed, the NTM must kill all processes which are children of the dead process. The children processes must not be given the option of continuing or dying; they must be killed.

c. The DRS will communicate with other processes via messages which are guaranteed to be delivered. The NTM must provide this facility in an efficient manner, and must include a mechanism to properly handle the node-dropping/node-returning/node-isolated problems.

3.1.3 Design/Implementation Differences

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

The only difference is the way in which CS-ES transformers are invoked by user APs at run-time. The design indicates that they are invoked via the Distributed Request Supervisor (DRS), but the Precompiler software generates code into the user APs to invoke them directly. This is possible because every CS-ES transformer must run on the same host computer as its user AP. Consequently, the DRS (and the NTM) are not needed. This was not forseen when the design was prepared and time did not permit the design to be changed later. This difference affects the PRE10 and PRE15 modules of the Precompiler as well as the DRS. The design of PRE10 should indicate that code is generated into the user AP source program to invoke each CS-ES transformer directly as a subroutine, rather than via the DRS and a local subroutine caller (LSC). The design of PRE15 should not indicate that LSCs are generated containing code to invoke CS-ES transformers. The design of the DRS should not indicate that DRSs are involved in invoking CS-ES transformers.

3.2 Detailed Functional Requirements

The following subsections respectively document each of the Distributed Request Supervisor major functions identified in Section 1.2.

3.2.1 Function DRS1: Initiate/Resume Subtransaction Processing

This function directs appropriate Request Processors to begin or resume processing. The Request Processors were built by the Precompiler CI and are co-located with their target databases.

3.2.1.1 Inputs

Inputs to this function are:

o The program ID (PID) and runtime parameters for each Request Processor which is to be activated for the NDML request. Four data items are input for each Request Processor to be activated: the PID, a code indicating whether to use a subroutine call or the NTM, a string containing the corresponding run-time parameters, and the length of the string.

   Run-time parameters are to be applied by the Request Processors to the subtransactions. These parameters are the values of the COBOL variables that were part of the NDML query. The first variable must contain the CASE statement number generated by the Precompiler.

o Responses from the NTM as a result of the DRS START-REMOTE requests for the RPs. These responses will contain the Logical Channel ID and the local process ID for the initiated RPs.

o Response from the Request Processor as a result of a START-LOCAL request. These responses will contain the Logical Channel ID assigned.

o Join Query Graph corresponding to the NDML request being processed.

o An attribute pair list. This list contains information concerning the join fields of the join edges of the Join Query Graph. This list will not be present for an update request.

o A Result Field Table. This table contains information about the result and join attributes of the query. This list will not be present for an update request.

o A Request Processor Information Table (RPIT) (see 3.2.2.4).

The first input is received from the Generate Request Processor function (PRE9) of the Precompiler CI. The other inputs are from the Decompose CS NDML function (PRE5) of the Precompiler CI.

A join query graph (JQG) corresponds to a CS NDML verb. Each node of a JQG represents an intermediate relation that will result from processing a single subtransaction, which accesses one database. Each edge of a JQG represents an inter-database join, union, or outer join operation between two relations. The set of edges represents the join, union, and outer join operations that in combination

will result in the response to the CS NDML transaction. The
format of the JQG is a table, with an entry for each edge of
the graph. Each entry contains the following information:

        rel1 rel2 edge-type attr-ptr PID1 PID2

where

        rel1   =   one of the edge nodes (PID of the RP that
                                         will create rel1)
        rel2   =   the other node (PID of the RP that will
                                         create rel2)
        edge-type =   (join)   = 4
                        (union)   = 5
                        (outer join) = 7
                        (deleted)   = *
        attr-ptr   =   A pointer into the attribute pair list.
                        It is null if the edge type is UNION.

        The format of the attribute pair list (APL) is a group of
linked lists of attribute pairs, one linked list for each edge
                                    of the JQG. Each
                                    entry in each list
                                    contains the
                                    following:

        rel1 rel2 attr1 comparison-op attr2 link

where

        rel1   =   the name of one edge node
        rel2   =   the name of the other edge node
        attr1   =   the Attribute Use Class number (AUC) of the
                        attribute of rel1 which is participating in
                        the join with attr2.
        attr2   =   The AUC of the attribute of rel2 which is
                        participating in the join with attr1.
        link   =   a pointer to the next entry in the list. (A
                        join can have more than one join field
                        pair). The field is null if there are no
                        more entries in the list.
comparison-op =   the operator (=, <, >, <=, >=, !=) used to
                        compare two attributes.

        Each entry in the Result Field Table (RFT) has the
following format:

        rel attr type size nd PID is-ptr

where

        rel   =   the name of the relation (subtransaction)
                        that contains the field
        attr   =   the Attribute Use Class number (AUC) of the
                        field

type	=	the type of the field (alphabetic, numeric, etc.)
size	=	the size, in bytes, of the field
nd	=	the number of decimal places maintained in the field
PID	=	not used
is-ptr	=	not used

3.2.1.2 Processing

This function starts the processing of the subtransactions that comprise a distributed database access or update. The following steps are performed by this function.

1. Initialize the Relation Information Table (RIT). The format of this table is described in Section 3.2.2.4. An entry is placed in the RIT for each relation to be constructed or accessed by a Request Processor.

2. Replace the Rel1 and Rel2 values in the JQG with an index number into the RIT, which corresponds to the entry for the appropriate relation.

3. Replace the Rel values in the RFT with the corresponding index number into the RIT if this request is an access request.

4. If the RPIT (Request Processor Information Table) (implemented as the QIT, see 3.2.2.4) has not been initialized for the specific instance of the AP, or if an existing RPIT has been set to "uninitialized" by the appearance of an NDML recovery unit termination request, the RPIT is now initialized by establishing such a table with one entry per RP name contained in the input message stream. If the RPIT had already been established, continue.

5. If Step 4 initialized the RPIT:

    First, initiate each RP with RP-call-type "R" in the QIT by issuing a STARTREMOTE message to the NTM. Then, initiate each RP with RP-call-type = "L" by calling it as a subroutine. The information given to the subroutine or the NTM will include the RP PID and the runtime parameters for each RP. The CASE statement number passed to the RP during this initiation phase is to be zero.

    The NTM will return, for each remote RP initiated, the Logical Channel ID and the local process ID. The Logical Channel ID is to be placed into the corresponding entry in the RIT.

The local process ID for each remote RP is to be placed into the corresponding entry in the RPIT.

The Host ID of the corresponding entries in the RIT are to be set to the Host ID of the hosts upon which each local or remote RP is running.

The status fields in the RIT are to be set to "BUSY".

If Step 4 did not initialize the RPIT:

Update the RPIT with any RPs that have a PID in the input message but do not currently appear in the RPIT. For those RPs added to the QIT, perform the functions stated in the above description for a newly-initiated RPIT.

Processing a subtransaction involves performing local restricts (a.k.a. selects), projects, and single-database joins. These operators have been translated to the DML appropriate for the local DBMS by the NDML/Generic DML Transformer function (PRE7) and the RP Generator function (PRE9) of the Precompiler CI. The local result relations should contain only join attributes and (final) result attributes.

3.2.1.3   Outputs

The outputs of this function are:

o   STARTREMOTE messages that activate or resume remote Request Processors. These messages contain the run-time parameters of the subtransaction.

o   An initialized RIT, if the request is an access request.

o   The modified JQG, if the request is an access request.

o   An initialized or modified RPIT.

3.2.2   Function DRS2:   Schedule Stages

This function iteratively determines the sequence in which intermediate relations are combined to form the result of a distributed database access. The sequence of join/union/outer join activities may include both parallel and serial processing.

3.2.2.1   Inputs

Inputs to this function are:

o   Join Query Graph corresponding to the NDML request being scheduled.

o   Result Field Table, if the request is an access request.

o   CS-ACTION-LIST, if the request is an access request.

o   ENDLOCAL messages

o   ENDREMOTE status parameter

o   ENDJOIN, ENDUNION, ENDOUTERJOIN messages

o   ENDFILESEND MESSAGES

o   Request Processor Information Table

The Join Query Graph (JQG) and Result Field table (RFT) inputs are also inputs to function DRS1 which modified them, and are described in section 3.2.1.1.

The CS-ACTION-LIST (CSAL) is a list of the attributes which will comprise the result relation. The order of this list is the order of the attributes in which the CS/ES Transformer will expect the final results to be. The format of the CSAL is the following:

ent-class auc workptr type size nd

where

ent-class	=	not used
auc	=	Attribute Use Class of the attribute
workptr	=	not used
type	=	not used
size	=	not used
nd	=	not used

ENDREMOTE messages are issued by the Request Processors and contain information about the intermediate relations that result from local processing. They arrive on the same logical Channel ID as was assigned when the Request Processor was initiated or via a parameter if the RP was called as a subroutine. ENDREMOTE indicates that processing of a subtransaction has been completed. The form of an ENDREMOTE message is:

ENDREMOTE length filename

where

length	=	the number of tuples in the resultant relation
filename	=	the name of the file on which the output was stored

ENDJOIN, ENDUNION, and ENDOUTERJOIN messages are issued by the Aggregator CI and contain information about the relations that result from combining intermediate relations. As in ENDREMOTE messages, they arrive on the same logical channel as was assigned when the Aggregator was invoked or via subroutine parameters.

ENDJOIN indicates that processing of an inter-database join has been completed. The form of an ENDJOIN message is:

ENDJOIN   length filename

where

length   = the number of tuples in the resultant
            relation
filename = the name of the file on which the output
            was stored

ENDUNION indicates that the processing of an inter-database union has been completed. The form of an ENDUNION message is:

ENDUNION   filename

where

filename = the name of the file on which the output was
            stored

Length is not available since one of the input operands is not read or counted; it is merely appended to by the UNION aggregator. The DRS knows a priori the length of each UNION operand and hence can sum the two to arrive at a total length of the resultant relation.

ENDOUTERJOIN indicates that the processing of an inter-database outerjoin has been completed. The form of an ENDOUTERJOIN message is:

ENDOUTERJOIN length filename

where

length   = the number of tuples in the resultant
            relation
filename = the name of the file on which the output
            was stored

ENDFILESEND messages are sent by the File Transfer process to acknowledge that a file has been sent as requested.

3.2.2.2 Processing

This function receives information about the results of processing of intermediate results.

The following describes the algorithm used by this module to control the runtime evaluation of a request. The general strategy is to break the request into stages, execute the stages serially, but execute the components of each stage in parallel. The first stage is comprised of all the request processors. Subsequent stages consist of file transfer requests, join requests, union requests, and/or outer join requests. Step 4, described below, determines which requests comprise subsequent stages.

Step 1.   Initialize Scheduler Tables

Create or update the following Performance Information Tables (PITs):

a.   Read the Transmission Cost Table (TCT) from a file. There is one TCT file at each site. The format of this table is described in Section 3.2.2.4. Implementation Note: this table is compiled into the DRS.

b.   For each entry in the RIT, calculate the width of the relation to be constructed by the corresponding request processor. This is calculated by examining the corresponding entries in the RFT. Enter each width into the appropriate RIT entry. Add 1 for each coresponding RFT entry to account for the null flag.

c.   Initialize the Cost Information Table (CIT). Each entry in the CIT corresponds to a candidate union, join or outer join action. The format of this table is described in Section 3.2.2.4. For each entry in the JQG, there will be two entries in the CIT. The first entry will have rel1 as the source relation and rel2 as the dest relation, and the other entry will have rel2 as the source relation and rel1 as the dest relation. For each entry placed in the CIT, set the source and dest fields as just described, set the i-p field to null, and set the edge-id field to the appropriate index into the JQG.

Step 2.   Process Incoming Messages

This step processes reply messages sent by the Request Processors, Aggregators and return parameters from local subroutines and aggregators. The Logical Channel ID, which is a part of each message or parameter is used to locate the entries in the RIT and CIT which correspond to the relation created by the process issuing the message.

a.   Process ENDREMOTE messages

o   Update the RIT entry corresponding to this
message. Set the length field with the
value returned in the message, and set the
status field to FREE. Store the results file
name in the RIT entry.

o   Scan the RIT. If there are any entries
with the status field equal to Busy,
go to Step 2, else go to Step 3.

b.   Process ENDJOIN, ENDUNION and/or ENDOUTERJOIN
messages

o   Update the RIT entry corresponding to this
message. Set the length field with the
value returned in the message, and set the
status field to FREE. Store the results
file name in the RIT entry.

o   Remove the corresponding CIT entry from the
CIT.

o   Scan the CIT. If there are any entries
with an i-p value of T or P, go to
Step 2, else go to Step 3.

c.   Process ENDFILESEND messages

o   Locate the RIT and CIT entries associated
with the logical channel ID of the message.
Store the results file name in the RIT
entry.

Go to Step 4.c.

Step 3.   Calculate Costs

This step removes duplicate entries in the CIT, and
calculates the cost for each remaining entry. If the
CIT is empty, then Function DRS2 is completed.

a.   Remove all entries in the CIT which have the
same source and dest relation as a previous
entry.

b.   If the CIT is empty, the RP operations requested
for the NDML request have been completed, with
the possible exception of an Aggregator step for
the termination of a recovery unit, the RPs that
have been operating on behalf of the AP must now
be stopped. Each of the RPs has, by this time,
executed a local recovery unit termination as a
result of the NDML request itself. The DRS must

now request that the NTM terminate the RPs
indicated by the entries in the RPIT. Therefore,
for each entry in the RPIT, the DRS will send a
termination request to the NTM, indicating the
host-id and the local process id that is to be
terminated. (implementation has shown this step
to be incorrect).

c.   Calculate the cost for each remaining entry by
     multiplying the length of the source relation by
     the width of the source relation by the trans-
     mission cost factor. The lengths and widths are
     obtained from the RIT, and the transmission cost
     factor is obtained from the TCT. Put the cost
     value in the corresponding entry in the CIT.

Step 4.   Process Join, Union, and Outer Join Edges

This step selects which join, union, or outer join
is to be performed next, updates the PITs
appropriately, sends FILE-TRANSFER messages, and
invokes Aggregators to perform the selected join,
union, or outer join.

a.   Select the next join, union, or outer join to
     process.

     o   Select the next lowest cost entry in the
         CIT; call that entry c-i.

     o   If the edge-type is JOIN or OUTER JOIN
         (found in JQG) AND either the source or
         dest relation appears elsewhere in an
         CIT entry which has an edge type of UNION,
         then go to Step 4.1a

     o   If the status of either the source or dest
         relation (found in RIT) is not FREE, then
         go to Step 4.a.

b.   Update the PITs

     o   Change the status field of the RIT for the
         source and dest relation entries to BUSY.

     o   Remove the other entry in the CIT with the
         same edge-id.

     o   Add a new entry to the RIT, which
         corresponds to the results of the join,
         union, or outer join to be performed. Set
         the status to BUSY.

o   Add an RFT entry for each attribute which
will appear in the result relation. All
fields of both the source and dest op-
erands, except join or outer join fields,
will appear in the result relation. For
each join or not-in-set field, scan the
APL for an entry which contains it. If it
appears in the APL entry other than the
current one, then it must appear in the
result relation. Calculate the width of
the resultant relation, and place the
value in its RIT entry.

o   Change all Rel fields in the JQG, CIT, and
APL whose value equals either the source or
dest relation RIT index to the RIT index
of the result relation.

c.   Send FILE-TRANSFER, JOIN, UNION and/or OUTER JOIN
messages

o   If the source relation is not at the same
site as the dest relation, send the source
file. Update the appropriate RIT and CIT
entries with Logical Channel ID associated
with the file transfer, and change the Host
ID field appropriately. Set the i-p field
in the current CIT entry to T. Go to Step
4.a.

o   Initiate the appropriate Aggregator to
perform the join, union, or outer join
operation, as designated by the edge-type.
If the Aggregator is to run on the same
node as the user AP, initiate
the Aggregator via direct call; otherwise,
initiate it via the NTM. Update the
appropriate RIT and CIT entries with the
logical channel ID associated with the
join. Set the i-p field of the current C"T
entry to P. Set the Host-id field in the
RIT appropriately. The formats of the JOIN
and UNION messages are described in Section
3.2.2.3; that of the OUTER JOIN message, in
AGG1.

o   If we got to Step 4.c as a direct result of
an ENDFILESEND message, go to STEP 2,
else go Step 4.

3.2.2.3   Outputs

The outputs of this function are the FILE-TRANSFER, JOIN,
UNION, and NOT-IN-SET messages, and NTM requests for process
termination.

a.   A FILE-TRANSFER message has the following format:

>   FILE-TRANSFER stage-id from-site to-site rel1
>   rel2

where

from-site	=	the current site location of the relation
to-site	=	the destination site for the relation
rel1	=	the file name of the relation to be sent
rel2	=	the file name of the relation on the to-site

b.   A JOIN message has the following format:

>   JOIN rel1 rel2 APL rel1-rft rel2-rft
>   result-rft

where

rel1	=	the file name of one of the relations to be joined
rel2	=	the file name of the other relation to be joined
APL	=	An attribute pair list of the join fi:'d pairs for this join
rel1-rft	=	An RFT for the fields of Rel1
rel2-rft	=	An RFT for the fields of Rel2
result-rft	=	An RFT for the fields of the result relation

c.   A UNION message stage has the following format:

>   UNION rel1 rel2

where

rel1	=	the file name of one of the relations to be unioned
rel2	=	the name of the other relation to be unioned

d.   See AGG1 for the format of the OUTER JOIN message.

e.   An NTM message requesting process termination has the following format:

>   STOP host-id RP-process-id

where

host-id	=	id of the host upon which the RP is running
RP-process-id	=	local process id of the RP that is to be stopped

3.2.2.4   Internal Data Requirements

Working data requirements include the following performance information tables (PITs):

a.   Transmission Cost Table (TCT):

The TCT contains transmission rates between each pair of application clusters and has the following format:

ac-1 ac-2 cost

This table should already exist in a file at each site. This has been implemented as a host specific COBOL copy member at each node.

b.   Relation Information Table (RIT):

The RIT contains information about each relation in the transaction, with the format:

rel-id length width status log-ch Host-id

where

rel-id	=	The file name where the relation resides
length	=	number of tuples in the relation
width	=	size of one tuple, in bytes
status	=	the present status of the relation (FREE or BUSY)
subtrans	=	the subtrans id of the relation

c.   Request Processor Information Table (RPIT):

The RPIT contains information about each RP that is currently operating on behalf of the AP. It has the format:

RP-host-id RP-process-id RP-call-type
RP-commit-roll-status RP-updates-attempted
RP-updates-Succ RP-begin-time RP-end-time

where

RP-call-type	= code indicating whether the RP is activated via the NTM (type

                                     = "R" for remote) or is called
as a subroutine (type = "L"
for local)

RP-host-id	= id of the host upon which the RP is running
RP-process-id	= process id of the RP on the specific host
RP-commit-roll-status	= the status of the commit or rollback performed by the RP (if any)
RP-updates-attempted	= the number of database updates attempted by the RP
RP-updates-succ	= the number of database updates that were successfully completed by the RP
RP-begin-time	= the clock time that the RP commit or rollback was started (if any) (used for RP logging purposes)
RP-end-time	= the clock time that the RP commit or rollback ended (if any) (used for RP logging purposes)

d.   Cost Information Table (CIT):

The CIT contains information regarding all possible
inter-database joins and unions, with the format.

      i-p edge-id source dest cost log-ch

where

i-p	=	a flag to mark the join as P (in progress) or T (file heirs transferred)
edge-id	=	the index of the particular edge into the JQG
source	=	the RIT index of the relation that will be transmitted
dest	=	the RIT index of the other relation
cost	=	the cost of transmitting the source relation
log-ch	=	the logical channel ID associated with the Aggregator process which is performing the join, union, or outer join.

The cost value is determined by multiplying the
length of the source relation from the RIT, times the
width of the source relation from the RIT, times the

appropriate cost formula from the TCT.

e.   Result Field Table (RFT):

The RFT is originally input from the Decompose CS NDML function (PRE6), but must be updated to keep track of result attributes as joins are processed.

3.2.3   Function DRS3:   Transfer Final Results

This function will, if necessary, move the final results file to the node on which the user AP and DRS is operating.   It is executed only for access requests (actions S, 1, 2, or K).

3.2.3.1   Inputs

The inputs to this function are:

o   The host id on which the user's AP is running

o   The current host on which the final results of the query or aggregation are stored.

o   The name of the final results file.

The first input is hardwired into each host specific version of the DRS.   The second is obtained from the only entry in the RIT marked as free.   The third input is also found in the same entry of the RIT.

3.2.3.2   Processing

o   Compare the host of the user AP and the host of the final results.   If they differ, issue a FILE TRANSFER message to move the file.

o   Wait for the reply from FILESEND.

o   When the FILESEND is complete, delete the final results file that was on the other host

3.2.3.3 Outputs

The outputs of this function are the following:

o   The file send message, if required, to move the final result

o   A transaction complete message, which is returned as a parameter to the application process requesting the transaction.   This message includes the file name of the results.

3.3 Special Requirements

Principles of structured design and programming will be adhered to.

3.4 **Human Performance**

Not applicable.

3.5 **Database Requirements**

Not applicable.

3.6 **Adaptation Requirements**

The system will be implemented at the ICAM IISS Test Bed site. The first Distributed Request Supervisor process will be implemented on the VAX VMS host.

SECTION 4

QUALITY ASSURANCE PROVISION

Among the tests that should be incorporated into the software are:

a.   input data checks

b.   interface data checks, i.e. tests to determine validity of data passed from calling routine

c.   database verification

d.   operator command checks

e.   output data checks

Not all tests are required in all routines, but error checking is an essential part of all software.

The CI quality assurance provisions must consist of three levels of test, validation and qualification of the constructed application software.

a.   The initial level can consist of the normal testing techniques that are accomplished during the construction process. They consist of design and code walk-throughs, unit testing, and integration testing. These tests will be performed by the design team which will be organized in a manner similar to that discussed by Weinberg in his text on software development team organization (THE PSYCHOLOGY OF COMPUTER PROGRAMMING, Van Nostran Reinhold, 1971). Essentially a team is assigned to work on a sub-system or CI. This approach has been referred to as "adaptive teams" and "egoless teams." Members of the team are involved in the overall design of the sub-system; there is better control and members are ex-posed to each other's design. The specific advantage from a quality assurance point is the formalized critique of design walk-throughs which are a preventive measure for design errors and program "bugs." Structured design, design walk-throughs and the incorporation of "antibugging" facilitate this level of testing by exposing and addressing problem areas before they become coded "bugs."

b.   Preliminary qualification tests of the CI are per-formed to highlight the special functions of the CI from an integrated point of view. Certain functional requirements may require the cooperative execution of one or more modules to achieve an intermediate or special function of the CI. Specific test plans will

be provided for the validation of this type of functional requirement including preparation of appropriate test data. (Selected functions from 3.2 must be listed).

c. Formal Qualification Tests will verify the functional performance of all the modules within the CI as an integrated unit that accept the specified input, perform the specified processes and deliver the specified outputs. Special consideration must be given to test data to verify that proper interfaces between modules have been constructed.

SECTION 5

PREPARATION FOR DELIVERY

The implementation site for the constructed software will be the ICAM Integrated Support System (IISS) Test Bed site. The required computer equipment will have been installed. The constructed software will be transferred to the IISS system via appropriate storage media.