

AD-A181 711

INTEGRATED INFORMATION SUPPORT SYSTEM (IIS) VOLUME 5

1/1

COMMON DATA MODEL 5 (U) GENERAL ELECTRIC CO

SCHENECTADY NY PRODUCTION RESOURCES CONSU

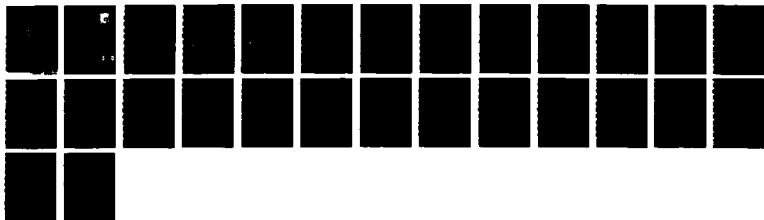
M LOOMIS

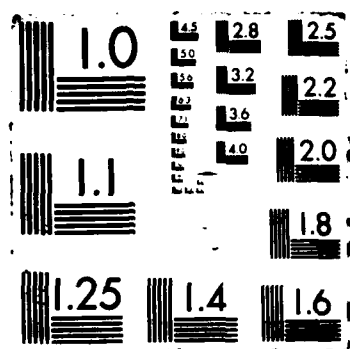
UNCLASSIFIED

01 NOV 85 DS-620141320

F/G 12/5

NL

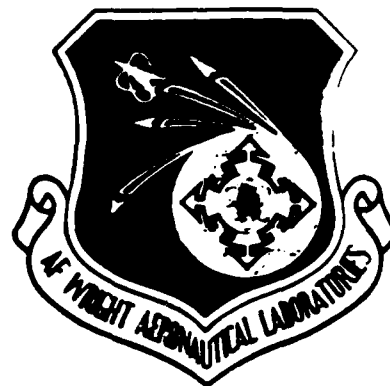




DTIC FILE COPY

AFWAL-TR-86-4006
Volume V
Part 28

AD-A181 711



INTEGRATED INFORMATION
SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 28 - Data Aggregators Development Specification

General Electric Company
Production Resources Consulting
One River Road
Schenectady, New York 12345

Final Report for Period 22 September 1980 - 31 July 1985
November 1985

Approved for public release; distribution is unlimited.

MATERIALS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AFB, OH 45433-6533

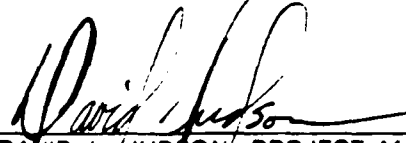
DTIC
ELECTE
JUN 30 1987
S D
A

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; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

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



DAVID L. JUDSON, PROJECT MANAGER
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

5 Aug 1986

DATE

FOR THE COMMANDER:



GERALD C. SHUMAKER, BRANCH CHIEF
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

7 Aug 86

DATE

"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 AFWAL/MLTC, W-PAFB, OH 45433 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.	
7a DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-86-4006 Vol V. Part 28	
6a NAME OF PERFORMING ORGANIZATION General Electric Company Production Resources Consulting	6b OFFICE SYMBOL <i>(if applicable)</i> AFVAL/MLTC	7a NAME OF MONITORING ORGANIZATION AFVAL/MLTC	
6c ADDRESS (City, State and ZIP Code) 1 River Road Schenectady, NY 12345		7b ADDRESS (City, State and ZIP Code) VPAFB, OH 45433-6533	
8a NAME OF FUNDING/SPONSORING ORGANIZATION Materials Laboratory Air Force Systems Command, USAF	8b OFFICE SYMBOL <i>(if applicable)</i> AFVAL/MLTC	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-80-C-8155	
8c ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, Ohio 45433		10. SOURCE OF FUNDING NOS.	
		PROGRAM ELEMENT NO. 78011F	PROJECT NO. 7500
		TASK NO. 62	WORK UNIT NO. 01
11 TITLE (Include Security Classification) (See Reverse)			
12. PERSONAL AUTHOR(S) Loomis, Mary			
13a TYPE OF REPORT Final Technical Report	13b TIME COVERED 22 Sept 1980 - 31 July 1981	14 DATE OF REPORT (Yr., Mo., Day) 1985 November	15 PAGE COUNT 29
16 SUPPLEMENTARY NOTES ICAN Project Priority 8301		The computer software contained herein are theoretical and/or references that in no way reflect Air Force-owned or developed computer software.	
17. CREAT. CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB GR	
1306	0805		
19 ABSTRACT (Continue on reverse if necessary and identify by block number) The Common Data Model Processor (CDMP) is a mechanism by which application programs can retrieve and update data without knowing where or how the data are stored. An application program poses requests to the CDMP, which processes those requests against the databases in which the relevant data are stored and then returns the results to the application program. The Neutral Data Manipulation Language (NDML) is the means for posing requests to the CDMP. Some NDML requests involve data that are stored in more than one database. One component of the CDMP, call the Aggregator, combines the data from each of these databases into a complete result for such a request. This development specification describes the functions, performance, environment, interfaces, and design requirements of the Aggregator.			
20 DISTRIBUTION AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐		21 ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL David L. Johnson		22b TELEPHONE NUMBER <i>(Include Area Code)</i> 813-255-8876	22c OFFICE SYMBOL AFVAL/MLTC

11. Title

Integrated Information Support System (IISS)
Vol V - Common Data Model Subsystem
Part 28 - Data Aggregators Development Specification

A S D 86 1440
17 Jul 1986



Approved For	
IN GRAMS	☒
DATA TAP	☐
Unrestricted	☐
Restricted	☐
By	
Date	
Availability Codes	
Avail and/or	
Special	
Test	
AI	

PREFACE

This development specification covers the work performed under Air Force Contract F33615-80-C-5155 (ICAM Project 6201). This contract is sponsored by the Materials Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Gerald C. Shumaker, ICAM Program Manager, Manufacturing Technology Division, through Project Manager, Mr. David Judson. The Prime Contractor was Production Resources Consulting of the General Electric Company, Schenectady, New York, under the direction of Mr. Alan Rubenstein. The General Electric Project Manager was Mr. Myron Hurlbut of Industrial Automation Systems Department, Albany, New York.

Certain work aimed at improving Test Bed Technology has been performed by other contracts with Project 6201 performing integrating functions. This work consisted of enhancements to Test Bed software and establishment and operation of Test Bed hardware and communications for developers and other users. Documentation relating to the Test Bed from all of these contractors and projects have been integrated under Project 6201 for publication and treatment as an integrated set of documents. The particular contributors to each document are noted on the Report Documentation Page (DD1473). A listing and description of the entire project documentation system and how they are related is contained in document FTR620100001, Project Overview.

The subcontractors and their contributing activities were as follows:

TASK 4.2

Subcontractors

Role

Boeing Military Aircraft
Company (BMAC)

Reviewer

D. Appleton Company
(DACOM)

Responsible for IDEF support,
state-of-the-art literature
search

General Dynamics/
Ft. Worth

Responsible for factory view
function and information
models

DS 620141320
1 November 1985

Subcontractors

Role

Illinois Institute of
Technology

Responsible for factory view
function research (IITRI)
and information models of
small and medium-size business

North American Rockwell

Reviewer

Northrop Corporation

Responsible for factory view
function and information
models

Pritsker and Associates

Responsible for IDEF2 support

SofTech

Responsible for IDEFO support

TASKS 4.3 - 4.9 (TEST BED)

Subcontractors

Role

Boeing Military Aircraft
Company (EMAC)

Responsible for consultation on
applications of the technology
and on IBM computer technology.

Computer Technology
Associates (CTA)

Assisted in the areas of
communications systems, system
design and integration
methodology, and design of the
Network Transaction Manager.

Control Data Corporation
(CDC)

Responsible for the Common Data
Model (CDM) implementation and
part of the CDM design (shared
with DACOM).

D. Appleton Company
(DACOM)

Responsible for the overall CDM
Subsystem design integration and
test plan, as well as part of
the design of the CDM (shared
with CDC). DACOM also
developed the Integration
Methodology and did the schema
mappings for the Application
Subsystems.

DS 620141320
1 November 1985

Subcontractors

Role

Digital Equipment
Corporation (DEC)

Consulting and support of the
performance testing and on DEC
software and computer systems
operation.

McDonnell Douglas
Automation Company
(McAuto)

Responsible for the support and
enhancements to the Network
Transaction Manager Subsystem
during 1984/1985 period.

On-Line Software
International (OSI)

Responsible for programming the
Communications Subsystem on the
IBM and for consulting on the
IBM.

Rath and Strong Systems
Products (RSSP) (In 1985
became McCormack & Dodge)

Responsible for assistance in
the implementation and use of
the MRP II package (PIOS) that
they supplied.

SofTech, Inc.

Responsible for the design and
implementation of the Network
Transaction Manager (NTM) in
1981/1984 period.

Software Performance
Engineering (SPE)

Responsible for directing the
work on performance evaluation
and analysis.

Structural Dynamics
Research Corporation
(SDRC)

Responsible for the User
Interface and Virtual Terminal
Interface Subsystems.

Other prime contractors under other projects who have
contributed to Test Bed Technology, their contributing
activities and responsible projects are as follows:

Contractors

ICAM Project

Contributing Activities

Boeing Military
Aircraft Company
(BMAC)

1701, 2201,
2202

Enhancements for IBM
node use. Technology
Transfer to Integrated
Sheet Metal Center
(ISMC)

DS 620141320
1 November 1985

<u>Contractors</u>	<u>ICAM Project</u>	<u>Contributing Activities</u>
Control Data Corporation (CDC)	1502, 1701	IISS enhancements to Common Data Model Processor (CDMP)
D. Appleton Company (DACOM)	1502	IISS enhancements to Integration Methodology
General Electric	1502	Operation of the Test Bed and communications equipment.
Hughes Aircraft Company (HAC)	1701	Test Bed enhancements
Structural Dynamics Research Corporation (SDRC)	1502, 1701, 1703	IISS enhancements to User Interface/Virtual Terminal Interface (UI/VTI)
Systran	1502	Test Bed enhancements. Operation of Test Bed.

TABLE OF CONTENTS

			<u>Page</u>
SECTION	1.0	SCOPE	1-1
	1.1	Identification	1-1
	1.2	Functional Summary	1-4
SECTION	2.0	DOCUMENTS	2-1
	2.1	Applicable Documents	2-1
	2.2	Terms and Abbreviations	2-2
SECTION	3.0	REQUIREMENTS	3-1
	3.1	Computer Program Definition	3-1
	3.1.1	System Capacities	3-1
	3.1.2	Interface Requirements	3-1
	3.2	Detailed Functional Requirements	3-1
	3.2.1	Function AGG1: Prepare Operands ...	3-1
	3.2.2	Function AGG2: Perform Union	3-4
	3.2.3	Function AGG3: Perform Join	3-5
	3.2.4	Function AGG4: Report Results	3-6
	3.2.5	Function AGG5: Perform NOT-IN-SET Selection	3-7
	3.3	Special Requirements	3-8
	3.4	Human Performance	3-8
	3.5	Database Requirements	3-8
	3.6	Adaptation Requirements	3-8
SECTION	4.0	QUALITY ASSURANCE PROVISIONS	4-1
SECTION	5.0	PREPARATION FOR DELIVERY	5-1

LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Title</u>	<u>Page</u>
1-1	AO of the CDMP Configuration Items	1-2

SECTION 1

SCOPE

1.1 Identification

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

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 (DRS), and the Aggregator. The scope of the CDMP and its configuration items is described in Figure 1-1 and the following narrative.

Common Data Model Processor (CDMP)

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

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

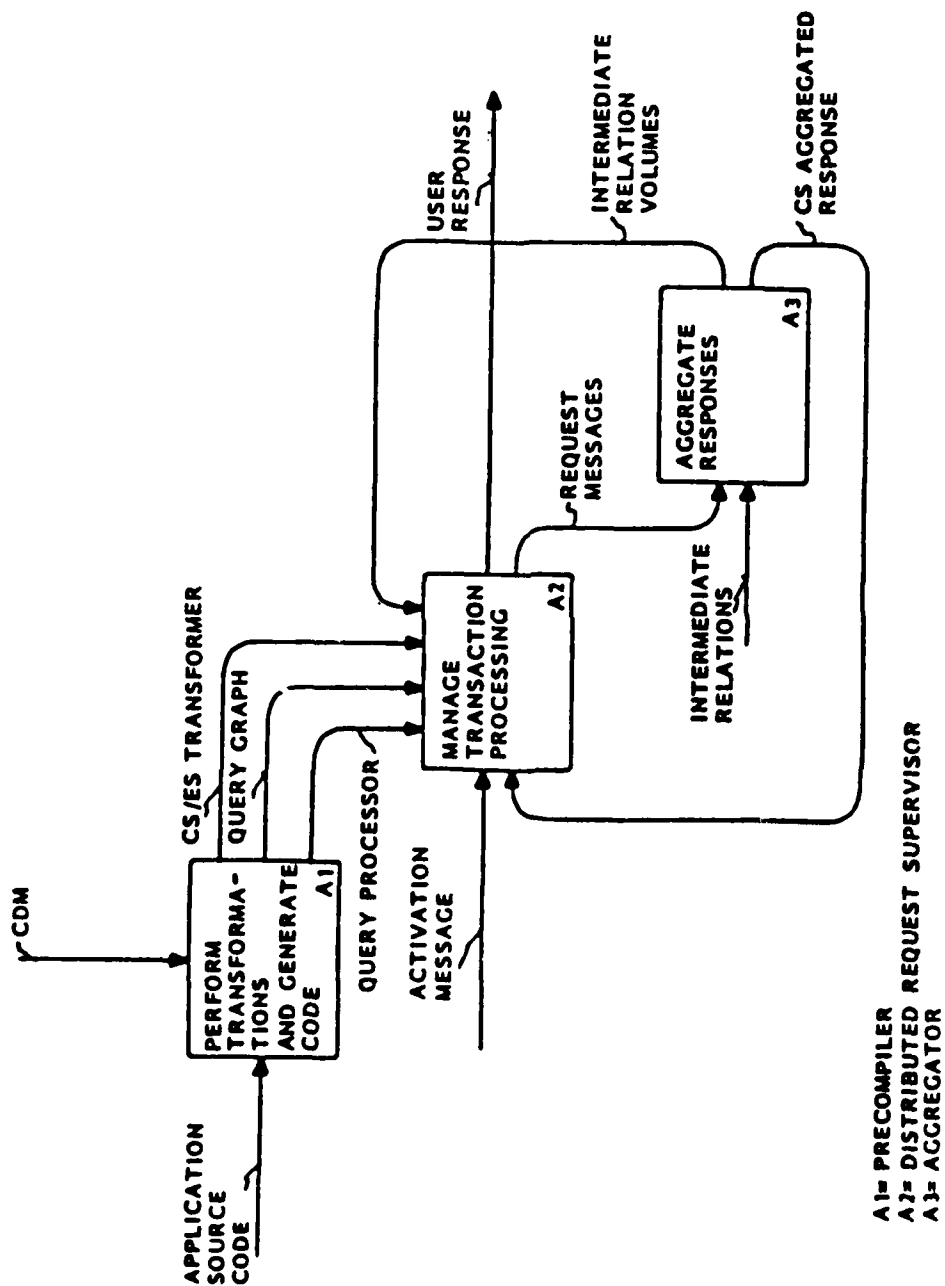


Figure 1-1. AO of the CDM Configuration Items

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

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

The DRS CI is responsible for coordination of the run-time activity associated with the evaluation of an NDML command. It is activated by the application program, which sends it the names and locations of the RPs to activate, along with run-time parameters which are to be sent to the RPs. The DRS activates the RPs, sending them the run-time parameters. The results of the RPs are stored as files, in the form of conceptual schema relations, on the host which executed the RP. Using the Join Request transmission cost information, and data about intermediate results, the DRS determines the optimal strategy for combining the intermediate results of the NDML command. It issues the appropriate file transfer requests, activates aggregators to perform join, union, and not-in-set operations, and activates the appropriate CS/ES Transformer program to transform the final results. Finally, the DRS notifies the application program that the request is completed, and sends it the name of the file which contains the results of the request.

The Aggregator CI is activated by the DRS. An instance of the Aggregator is executed for each join, union, or not-in-set performed. It is passed information describing the operation to be performed and the file names containing the operands of the operation. The DRS ensures that these files already exist on the host which is executing the particular Aggregator program. The Aggregator performs the requested operation, storing the results in a file whose name was specified by the DRS and is located on the host executing the Aggregator.

The CDMP provides the application programmer with important capabilities to:

1. Request database accesses in a non-procedural data manipulation language (the NDML) that is independent of the data manipulation language (DML) of any particular Data Base Management System (DBMS),
2. Request database access using a DML that specifies accesses to a set of related records rather than to

individual records, i.e., using a relational DML,

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

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

1.2 Functional Summary

The overall objective of this CI is to perform relation join, union, and not-in-set operations upon intermediate results of a multi-database transaction. It, along with the DRS, Request Processors, and local DBMS modules, performs the run-time evaluation of commands presented to them by application processes.

There are two Aggregator programs residing on each site containing a database, including the site containing the CDMP. They are identical except that one is invokable via an NTM message while the other is callable as a subroutine. The DRS determines which to use for each aggregation. This development specification describes the generic Aggregator CI, which is to be implemented as multiple Aggregator programs. The DRS causes an Aggregator program to be executed for each join, union, or not-in-set operation which is required to evaluate a particular NDML request. It is the DRS's responsibility to ensure that both operands to the operation already reside in files on the site where the operation is to be performed. When an Aggregator is activated, it is passed parameters which it needs to perform the operation. Included in these parameters are the file names of the operands and the file name where the result of the operation is to be stored. A logical channel ID is also a part of the activation message. This is a communications channel created between the Aggregator and the DRS. The reply message is sent by the Aggregator over this channel when the operation is completed.

The Aggregator sorts the files containing the operands appropriately, creates the file which will contain the resultant table, reads the operand files, performs the join, union, or not-in-set operation and stores the results in the resultant relation file. Finally, it deletes the operand files, sends a

DS 620141320
1 November 1985

completion message to the DRS which created it, and terminates.

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

- Function AGG1 Prepare Operands
- Function AGG2 Perform Union
- Function AGG3 Perform Join
- Function AGG4 Report Results
- Function AGG5 Perform Not-In-Set Selection

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) Aggregator.

Related ICAM Documents included:

UM620141001	<u>CDM Administrator's Manual</u>
TBM620141000	<u>CDM1, An IDEF1 Model of the Common Data Model</u>
UM620141100	<u>Neutral Data Definition Language (NDDL) User's Guide</u>
PRM620141200	<u>Embedded NDML Programmer's Reference Manual</u>
UM620141002	<u>ICAM Definition Method for Data Modeling (IDEF1-Extended)</u>
DS620141200	<u>Development Specification for the IISS NDML Precompiler Configuration Item</u>
DS620141310	<u>Development Specification for the IISS Distributed Request Supervisor Configuration Item</u>

Other references include:

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

Blasgen, M. W., Eswaran, K. P., "On the Evaluation of Quebies in A Relational Data Base System," IBM Research Report RJ1745, IBM Research Laboratory, San Jose, CA, April 1976.

Chamberlin, D. D., "Relational Data Base Management

Systems," Computing Surveys, Vol 8, No 1, March 1976.

Daniels, D., et. al., "An Introduction to Distributed Request Compilation in R*," IBM Research Report RJ3497, IBM Research Laboratory, San Jose, CA, June 1982.

Dejean, J.P., Test Bed System Development Specifications, General Electric Co., November 9, 1982.

Epstein, R., Stonebraker, M., and Wong, E., "Distributed Request Processing in a Relational Database System," Proceedings of the ACM SIGMOD International Conference, Austin, TX, June, 1978.

Lindsay, B. G., et. al., "Notes on Distributed Databases," IBM Research Report RJ2571, IBM Research Laboratory, San Jose, CA, July 14, 1979.

Williams, R., et. al., "R*: An Overview of the Architecture," IBM Research Report RJ3325, IBM Research Laboratory, San Jose, CA, December 12, 1981.

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

IS Internal Schema

DS 620141320
1 November 1985

NDML Neutral Data Manipulation Language

RP Request Processor

RFT Result Field Table

SDS System Design Specification

SECTION 3

REQUIREMENTS

3.1 Computer Program Definition

3.1.1 System Capacities

The capacity of the Aggregator is limited by the capacity of the file systems where the operands and resultant relations are stored and by the sorting capabilities of the computer. It is the responsibility of each site's system administrator to ensure that file system and memory allocation capacities are adequate.

3.1.2 Interface Requirements

3.1.2.1 Interface Blocks

This CI is the mechanism that aggregates results of logically related intermediate responses to a distributed database request. Its interfaces include input in the form of two types of requests and output in the form of a table (relation) and status responses.

3.1.2.2 Detailed Interface Definition

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

3.2 Detailed Functional Requirements

The following subsections document each of the Aggregator's major functions identified in Section 1.2.

3.2.1 Function AGG1: Prepare Operands

This function sorts the operands, which are stored in files, for the join, union, or not-in-set operation.

3.2.1.1 Inputs

The input to this function is the operation request, which is part of the message body sent by the DRS when the Aggregator is activated.

The format of a join request is the following:

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

where:

- rel1 - file name of the first operand.
- rel2 - file name of the second operand.
- result - file name where the result should be stored
- APL - pointer to attribute pair list (APL) entry containing information about the join fields.
- rel1-rft - pointer to Result Field Table (RFT) entry containing field format information for rel1.
- rel2-rft - pointer to an RFT entry for rel2.
- result-rft - pointer to an RFT entry for the result table.

The format of a union request is the following:

UNION rel1 rel2 result rft

The rel1, rel2, and result fields have the same meaning as that of the join request. There is only one RFT pointer, because the formats of both files and the result are identical.

The format of a NOT-IN-SET selection operation request is as follows:

NOT rel1 rel2 result APL rel1-rft rel2-rft

where:

rel1, rel2, and result have the same meaning as in the join request.

APL is similar to the APL for a join request but the

indicated fields are for the NOT-IN-SET selection.

rel1-rft is the RFT for rel1 and for the result.

rel2-rft is the RFT for rel2.

The format of the APL is a list of join attribute pairs.

Each entry in the list contains the following:

rel1 attr1 rel2 attr2 link

where:

rel1 = not used

attr1 = Attribute Use Class number (AUC) of an attribute from rel1

rel2 = not used

attr2 = AUC from rel2

link = not used

Each entry in the RFT has the following format:

rel attr type size ND PID is-ptr

where:

rel = not used

attr = AUC of the field

type = field type (alphabetic, numeric, etc.)

size = number of bytes in the field

ND = number of decimal places of the field

PID = not used

is-ptr = not used

3.2.1.2 Processing

This function sorts the operands appropriately. If the request type is union, then each operand is sorted in ascending sequence, starting with the leftmost field. If the request type is join or not-in-set, then each operand is sorted in ascending sequence, using its join field as the sort key.

It is assumed that the underlying operating system at each site where an Aggregator program executes provides sufficient sort facilities. Function AGG1 merely activates these facilities. If a site operating system does not provide sufficient sort facilities, then they will have to be developed, as a separate configuration item.

3.2.1.3 Outputs

The outputs of this function are the input files, sorted appropriately.

3.2.2 Function AGG2: Perform Union

This function performs a union operation on the input operands, and stores the results in a specified file.

3.2.2.1 Inputs

Inputs to this function are:

- The file names of the operands, sorted appropriately.
- The file name of the resultant relation

The operand files have been sorted by the AGG1 function, but the file names were left unchanged.

3.2.2.2 Processing

The algorithm used by this function is called a merge sort. Each operand file is opened, and the file which is to contain the resultant relation is created and opened. A record is read from each operand file, and the lexicographically lower record is chosen and written to the resultant relation file. Another record is read from the operand file of the record just written, and the process is repeated, until all records have been read. Then all three files are closed.

3.2.2.3 Outputs

There are two outputs of this function:

- The resultant relation file
- The number of records in the resultant relation file

3.2.3 Function AGG3: Perform Join

This function performs a join operation on the input arguments and stores the results in a specified file.

3.2.3.1 Inputs

Inputs to this function are:

- The file names of the operands, sorted appropriately
- The attribute pair list
- The three RFT's
- The file name of the resultant relation

All inputs come from the activation message. The operand files have been sorted by the AGG1 function, but the file names were left unchanged.

3.2.3.2 Processing

Both operand files are opened, and the resultant relation file is created and opened. For each record of the first operand, the second operand file is scanned for records whose values in its join fields match the values in the join fields of the record of the first operand. A record is created for each record of the second operand which meets the above conditions. The created record contains all the fields specified in the result RFT. Each created record is written to the resultant relation file.

Because both operands are sorted by their respective join fields, only one pass need be taken over each operand. However, all records of the second operand which participate in the join with a particular record of the first operand must be saved. This is necessary since the next record of the first operand may contain the same values in its join fields as the

previous record, thus requiring the saved records. If these join field values are not the same, then the required records from operand two have not been read yet, and the previously saved records can now be discarded. The resultant relation may contain duplicate records.

3.2.3.3 Outputs

There are two outputs of this function:

- The resultant relation file
- The number of records in the resultant relation file

3.2.4 Function AGG4: Report Results

This function removes the operand files, sends an ENDUNION, ENDJOIN or ENDNOT message to the DRS which initiated the action, and terminates execution of the Aggregator process.

3.2.4.1 Inputs

Inputs to this function are:

- file names of the operands
- logical channel id of the DRS process
- resultant file name
- number of records in the resultant relation

The first three inputs come from the activation message. The remaining input comes from either the AGG2, AGG3, or AGG5 functions.

3.2.4.2 Processing

This function makes calls to the operating system to remove the files storing the operands. It then constructs an ENDUNION, ENDJOIN, or ENDNOT message and sends it to the DRS processor which invoked the Aggregator process using the logical channel id or a subroutine parameter as appropriate. Finally, it causes the termination of the Aggregator process.

3.2.4.3 Outputs

The outputs of this module are the following:

1. Two file system calls to remove the operand files. These calls are site dependent.
2. An ENDUNION, ENDJOIN, or ENDNOT message. The message has the following format:

message length

where:

message = ENDJOIN, ENDUNION, OR ENDNOT

length = the number of records in the resultant relation

3.2.5 Function AGG5: Perform NOT-IN-SET Selection

This function performs a not-in-set selection on the first input file and stores the results in a specified file.

3.2.5.1 Inputs: Inputs to this function are:

- the file names of the operands, sorted appropriately
- the attribute pair list
- the RFTs of the operands
- the file name of the resultant relation

All inputs come from the activation message. The operands have been sorted by the AGG1 function but the file names were left unchanged.

3.2.5.2 Processing:

Both operand files are opened and the resultant relation file is created and opened. For each record of the first operand, the value of the selection field designated by the APL is compared with values in the selection field in the second operand; because both files are sorted by comparison fields, the second file need be searched only from the first occurrence of the value of a previous match to values greater than the value

sought. If no match is found, the record from the first file is copied to the result file; if a match is found, the record is not copied. The result record contains all the fields of the first operand.

3.2.5.3 Outputs

There are two outputs from this function:

- the resultant relation file
- the number of records in the resultant relation file

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 located at the General Electric facility in Schenectady, NY. The Aggregator processes will be implemented on the VAX VMS host and the IBM host.

SECTION 4

QUALITY ASSURANCE PROVISION

In preparation for describing requirements for quality assurance provisions, it is appropriate to define the terms "test" and "debug" which are often used interchangeably. "Testing" is a systematic process that may be preplanned and explicitly scheduled. Test techniques and procedures may be defined in advance and a sequence of test steps may be specified. "Debugging" is the process of isolation and correction of the cause of an error. To start with, the concept of "antibugging" is recommended in the construction of the software modules. In his text on software development (Techniques of Program Structure and Design, Prentice-Hall 1975), Yourdon defines antibugging as "the philosophy of writing programs in such a way as to make bugs less likely to occur, and when they do occur (which is inevitable), to make them more noticeable to the programmer and the user." That is, do as much error checking as is practical and possible in each routine.

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

1. input data checks
2. interface data checks, i.e., tests to determine validity of data passed from calling routine
3. database verification
4. operator command checks
5. 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.

1. 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 New York: Van Nostrand Reinhold, 1971). Essentially a team is assigned to work on a subsystem 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 subsystem, there is better control and members are exposed to each others 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."

2. Preliminary qualification tests of the CI are performed 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).
3. Formal Qualification Test 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 ensure verification that proper interface of modules has been constructed.

DS 620141320
1 November 1985

SECTION 5

PREPARATION FOR DELIVERY

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

GPO-748-061

END

7-87

Dtic