Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

RELATIONSHIPS IN A HETEROGENEOUS DISTRIBUTED DATABASE ENVIRONMENT

Jeffrey Raymond Horowitz

June 1989
CSD-890029

Relationships

In A

Heterogeneous Distributed Database

Environment

Jeffrey Raymond Horowitz
Informatics Software Systems
UCLA Computer Science Department
October 14, 1985

TABLE OF CONTENTS

* Part of this work was supported by the University of California MICRO Program, grant #P-5348-N83, with the cooperation and support of Informatics General Corporation.

Relationships In A HDDB Environment

ABSTRACT

A generalized database access path model is defined for the purpose of representing relationships between data entities in a Heterogeneous Distributed Database (HDDB) network. This data model, termed the Generalized Database Access Graph (GDAG), is one architectural component of a heterogeneous distributed database management system (HDDBMS). The GDAG is maintained by the HDDBMS as part of an intergrated data dictionary. It encompasses the capabiltity of modeling the access paths of the three major data models, relational, network, and hierarchical, via a common data independent notation. A further capability is the modeling of interdatabase relationships using an equivalent notation.

KEY WORDS AND PHRASES

Access Path Model, Data Independence, Entity-Relationship (ER) Model, Generalized Database Access Graph (GDAG), Heterogeneous Distributed Database, Interdatabase Relationship, Query/Data Manipulation Language Translation, and String Model.

Relationships In A HDDB Environment

1 INTRODUCTION

A heterogeneous distributed database (HDDB) is a set of two or more interrelated physical databases in which the underlying database management systems (DBMS) are different. For example, one physical database might be managed by IMS/DB, another by SQL/DS, and a third by a CODASYL DBMS.

HDDB management is a challenging problem induced over recent years by the proliferation of corporate databases. These databases are unable to automatically share information due particularly to the inherent database model and query/data manipulation language differences between different DBMS. Currently, difficult manual efforts must be employed to insure the accessiblity and correctness of the information in a HDDB.

A physical, or local, database of a HDDB may contain data which is related to or duplicated by another physical database. These relationship or content dependencies between physical databases are termed interdatabase relationships.

Interdatabase relationships pose a major problem if one is to automate the HDDB management. Queries upon the HDDB must be analyzed against interdatabase relationships to gather global results. Information accessed from one physical database might depend upon the information obtained from another. The existence of an entity in one physical database might depend upon the existence of another in a different physical database. Or, the contents of an entity might be spread across two or more physical databases.

2 OBJECTIVES

This research describes an access path model, named the Generalized Database Access Graph (GDAG), for relationships in a HDDB. Some elements of the GDAG were inspired by the Data Independent Access Model (DIAM) [1] and the Algebraic Access Graph (AAG) [6]. The access path model is used for query analysis by a HDDB Management System (HDDBMS). The access path specifications are maintained as part of an integrated data dictionary definition for the HDDB.

The access path model that follows is part of an ongoing research project [1, 2, and 7]. The goal of this project is to construct a HDDBMS which intercepts the database requests or queries from an application program, in the query or data manipulation language of a host DBMS, transforms these requests into the individual requests upon the physical databases, and translates the results into the form expected by the application program. The term data manipulation language (DML) shall be used to refer to any of the data access languages of the major types of DBMS: CODASYL DML; relational or SQL query languages; or IMS/DB DL/I. The terms "database request" and "query" will be synonymous.

3 HDDBMS LAYERED ARCHITECTURE

The HDDBMS architecture consists of five layers of database models. These are illustrated in Figure 1. An application program database view is defined using the data definition language of a host DBMS. This view is defined to the HDDBMS at the virtual layer (VL).

An application program query (DML or query command) enters the virtual layer and is transformed by the HDDBMS into a unified virtual layer (UVL) query. This layer is an Entity-Relationship (ER) [5] representation of the application program's virtual layer view.

The UVL query is then mapped into a unified global layer (UGL) query. The UGL is an ER conceptual representation of the entire HDDB. It represents the union of individual unified local layer (ULL) database views.

The UGL query is transformed into a set of one or more ULL queries and an access plan. A ULL definition exists for each physical database in the HDDB. Externally, a ULL definition of a physical database is an ER view of that database. Internally, ULL access path specifications exist for data within a single physical database and for each interdatabase relationship between two or more physical databases.

A ULL query is transformed into a Local Layer (LL) DBMS dependent query, and then sent to the local DBMS. The ULL queries are performed according to the precedence established by the access plan.

Once the results of the original query are obtained, the data is translated back through the layered architecture into the form expected by the application program. This involves both structural and data translation.

4 ACCESS PATH MODEL REQUIREMENTS

The access path model exists at the UGL and ULL of the HDDBMS. It is responsible for guiding the UGL query translation into the ULL queries and access plan. The access path model consists of an ordering of elementary operations at field level granularity which must be performed by the individual DBMS or by the global query coordinator.

The access path model is responsible for representing the elementary operations and the order of operation for data retrieval in an HDDB. UGL access path specifications exist for each relationship in the global ER view of the HDDB. These act as a basic floorplan for the ULL access paths. ULL access path specifications exist for each relationship in a physical database and for each interdatabase relationship between two or more physical databases.

Since these access path specifications are utilized by a HDDB global query analyzer and by DBMS dependent local query generators, they must be defined in a generalized form. That is, they cannot be dependent upon any particular DBMS. Hence, their structure cannot be DBMS dependent nor can their elementary operations contained within them. Yet, they must be capable of transformimg ULL queries into a local DBMS DML while maintaining the semantics of the original application program query.

As a further requirement, the modeling of local physical database relationships must be equivalent to the modeling of interdatabase relationships. This allows the global query analyzer to generate ULL queries independent of distribution. The responsibility of distribution is left to the access plan generator and the global query coordinator.

5 GENERALIZED DATABASE ACCESS GRAPHS (GDAG)

The Generalized Database Access Graph (GDAG) is defined below. First is a definition of the general GDAG structure. This is followed by a description of the notation used and the rules for GDAG traversal. Figure 2 represents a generic GDAG. The reader should view this Figure throughout this Section.

5.1 THE GENERAL GDAG STRUCTURE

A GDAG is a two dimensional labeled directed graph. The arc labels of a graph contain logical access criterion between nodes. The nodes, termed attribute nodes, contain data element, entity, or relationship specific information. These attribute nodes are termed global or local, depending on their context. A global attribute node simply identifies which portion of the UGL ER conceptual model the node applies to. A local attribute node contains information as to which global portion the node applies to as well as local database information.

The horizontal portion of a GDAG consists of a single linear string of global attribute nodes. Two special global attributes are defined for the horizontal direction. These are the graph entry and exit nodes.

Let R be an n-ary relationship between entities E1, E2, ..., and En. A horizontal string for R is of the form

```
entry --> Ej1 --> Ej2 --> ... --> Ejn --> R --> exit
```

where { j1, j2, ..., jn } is some permutation of { 1, 2, ..., n }. The nodes identified by entity names (Ej) are termed global entity nodes. The node identified by a relationship (R) is called a global relationship node. The arcs for the horizontal string are not labeled.

A vertical string is attached to each entity node and to the relationship node. A vertical string is a linear string of local attribute nodes attached by labeled arcs.

The local attribute nodes contain data element specific information that is either part of the global entity or relationship node for which the string is attached, or data element specific information that is required as a constraint to get the data elements of the global node. The data element specific information consists of the data element name as defined to a local database, its UGL ER attribute name, the physical database it belongs to, and the relation, segment, or record it is contained in.

The arc labels contain basic logical operations that must be completed to successfully traverse an arc. For example, an operation might be an equivalence of key fields, or an operation might be a continue operation (one that is always true).

5.2 NOTATION AT GDAG NODES

Relationships In A HDDB Environment

The following notation is used at the nodes contained in a GDAG. Notation is defined for both horizontal nodes and vertical nodes.

The notation at a horizontal node consists of an index value contained inside the node. This index value must be unique within the GDAG. The index value is used for identification purposes at vertical nodes. If the node does not denote entry or exit, it also contains a UGL identifier which is either the global entity or relationship it is associated to. This identifier is written above the horizontal node.

Vertical local attribute nodes are of three types. These are either entity, relationship, or constraint. Vertical entity nodes may only appear in a vertical string that is based off a horizontal global entity node. Similarly, vertical relationship nodes may only appear beneath a horizontal global relationship node. Vertical constraint nodes may appear in any vertical string of a local or interdatabase GDAG (ULL).

The notation at a vertical entity node consists of the following information. Located at the lower left portion of the node is the single character E. This denotes the node as a vertical entity node. Contained inside the node is the UGL attribute for which the node is associated. For ULL GDAG specifications, located at the lower right portion of the node is the physical database data element name. Located at the upper right hand portion is database identification information which is of the form:

global_entity_name (UGL)

or

index_value.DB_id.RECORD_id (ULL)

where global_entity_name is the name of an entity in the global ER view of the HDDB, index_value is an index contained in some horizontal node, DB_id identifies the physical database the data element exists at, and RECORD_id identifies the relation, segment, or record the data element exists in.

The notation at a ULL vertical relationship node differs from a ULL vertical entity node only in the node type. A vertical relationship node contains a single character R located at the lower left portion of the node. For a UGL vertical relationship node, the database identification information is excluded.

The notation at a vertical constraint node is similar to vertical entity and relationship nodes. Contained at the lower left portion of the node is a single character C. Always contained at the upper right portion of the node is the database identification information. Optionally, a UGL attribute name and physical database data element name are provided.

5.3 NOTATION ALONG GDAG VERTICAL ARCS

The vertical arcs of a GDAG are labeled with a logical access criterion. These are of three types: equivalence; continue; and database access path.

The equivalence operator, =, applies to the two connected local attribute nodes and states that the data elements of the two nodes must be equal in value for successful traversal of the arc. The equivalence operator is used mostly for defining relational joins or for establishing the link of an interdatabase relationship.

The continue operator, *, states that traversal of the arc is automatic. That is, its logical value is always true. This operator is used mostly for information gathering or stepping through data element by data element access where the data elements have been retrieved via an earlier vertical attribute node.

The database access path operator, ACP, states that traversal of the arc may proceed only if some database access path is successfully traversed. This operator is used with network and hierarchical database specifications. For a network database, the access path would be a set. For a hierarchical database, it would be a parent-child relationship.

5.4 GDAG TRAVERSAL RULES

The rules for traversing a GDAG are simple. They are as follows:

1. Traversal begins at the entry node.
2. Traversal ends successfully only upon reaching the exit node.
3. Horizontal arc traversal may proceed only when the vertical string of the current horizontal node has been successfully traversed.
4. Vertical arc traversal may proceed only if the logical operation along the arc yields a true value.
5. Vertical string traversal is successful upon reaching a vertical node which does not have an arc leaving it, and whose data element can be successfully retrieved.
6. Any other occurrence results in an unsuccessful GDAG traversal.

5.5 RELATIONSHIP DIRECTIONS

Since database access paths usually are not reversible, directions are associated to database relationships. For example, a relationship R between entities E1 and E2 can be viewed in two different ways: "how R relates E1 to E2"; and, vice-versa, "how R relates E2 to E1." Each of these views is termed a direction of R.

Generalizing to n-ary relationships, n! different directions exist. For example, a trinary relationship R between E1, E2, and E3 has these six directions:

1. How R relates E1 to E2 and E3;
2. How R relates E1 to E3 and E2;
3. How R relates E2 to E1 and E3;
4. How R relates E2 to E3 and E1;
5. How R relates E3 to E1 and E2; and
6. How R relates E3 to E2 and E1.

The notation used to denote relationship directions is a nesting of binary relationships. For the trinary relationship above, the notation would be:

1. R : E1-->(E2-->E3);
2. R : E1-->(E3-->E2);
3. R : E2-->(E1-->E3);
4. R : E2-->(E3-->E1);
5. R : E3-->(E1-->E2); and
6. R : E3-->(E2-->E1).

The parentheses are not really required. They do, however, suggest a mechanism for defining relationship directions algorithmically (via counting) and recursively (until an inner most binary relationship is encountered). Upon specifying all directions for a relationship, the parentheses may be discarded.

When considering binary relationships, two directions exist. In this case, each direction may be regarded as the inverse or reverse direction of the other.

Each UCL relationship direction requires a GDAG specification for each physical database containing an implementation of that relationship, and for each interdatabase relationship the UCL relationship is an instance of. The requirement of a GDAG for each relationship direction is a direct consequence of the non-invertibility of database access paths. This is especially true for hierarchical and network databases.

6 AN EXAMPLE HETEROGENEOUS DATABASE NETWORK

An example HDDB is used throughout the remainder of this document. It represents a Parts and Warehouse database. This database is described below. This description is followed by some example GDAG specfications for each physical database in the network, and for some of the interdatabase relationships.

6.1 THE GLOBAL ER VIEW OF THE EXAMPLE DATABASE

An ER view of the example database is illustrated in Figure 3. Two global entities exist. These are the Part (E1) and Warehouse (E2) entities. The Part entity contains three data elements. They are a unique key Part Number (P#) along with a Part Description (PD) and Part Classification (CL). The Warehouse entity consists of a unique key Warehouse Number (W#) and a Warehouse Description (WD).

Two global binary relationships are defined. One relationship (R1) represents the supply of Parts at Warehouses via the data element QTY (Quantity). The second (R2) is a recursive relationship which represents the construction of Parts by other Parts and the Quantity Used (QTY_USED) per construction. Since relationship R2 is a relationship of Parts to Parts, a futher classification of Parts as Assembly or Component Parts is required. These are denoted as E1a and E1c respectively.

Relationship R1 is useful for answering queries of the form

R1 : E1-->E2

"Given a Part, which Warehouses supply that Part, and what are the Quantities On Hand ?"

Inversely,

R1 : E2-->E1

"Given a Warehouse, which Parts does the Warehouse supply, and what are the Quantities On Hand ?"

Relationship R2 is useful for answering queries of the form

R2 : E1a-->E1c

"Given an Assembly Part, which Parts are components in the construction of the Assembly Part, and what is the Quantity Per Assembly ?"

Inversely,

R2 : E1c-->E1a

"Given a Component Part, which Assemblies require the Component Part for Assembly, and what is the Quantity Per Assembly ?"

6.2 THE PHYSICAL DATABASE DESCRIPTIONS

The example HDDB consists of three physical databases. One of these is a relational database (SQL/DS), another is network (CODASYL), and the third is hierarchical (IMS/DB). These are referred to as DB1, DB2, and DB3 respectively. The database definitions of each are illustrated in Figures 4, 5, and 6. The conceptual view of the SQL/DS database consists of the entire HDDB conceptual view. The CODASYL database consists of both entities E1 and E2 and relationship R1. The IMS/DB consists of entity E1 and relationship R2.

Several interdatabase relationships exist within the HDDB. DB1 and DB2 both contain the R1 relationship between Parts and Warehouses, and DB3 contains information about Parts which may be stored at DB1 and DB2. DB1 and DB3 each contain the R2 relationship of Parts to Parts, and DB2 contains information about Parts which may be stored at DB1 and DB3.

In the following some GDAG specifications for the HDDB are described. First the GDAGs for the global ER database view are presented. Next, the GDAGs for each physical database are illustrated. These are followed by some interdatabase relationship GDAGs. Since local GDAGs always apply to a single physical database, database identification information (DB_id) is left off the vertical attribute nodes in the examples. This is not true for the interdatabase relationship GDAGs.

6.3 GLOBAL GDAG SPECIFICATIONS

The only information contained in the Global GDAGs (UGL) is the global ER data attributes required to establish a global relationship since these GDAGs only represent a basic floorplan in guiding query and data translation.

Figures 7 and 8 illustrate the Global GDAGs for the ER view of the HDDB. Figure 7 contains the GDAGs for relationship R1 and Figure 8 contains them for relationship R2.

The information content required for relationship R1 is a P# (E1), a W# (E2), and the QTY (R1) . The information content required for relationship R2 is a P# (E1a), a P# (E1c), and the QTY_USED (R2). These data attributes exist in vertical strings relative to which portion of the relationship they apply to.

6.4 LOCAL GDAG SPECIFICATIONS

The following presents the Local GDAG specifications for each of the three example physical databases.

<u>SQL DB1</u>

Figures 9 and 10 illustrate the Local GDAGs (ULL) for the SQL/DS relational database DB1. Figure 9 contains the GDAGs for relationship R1 and Figure 10 contains them for relationship R2.

Relationship R1 in DB1 is represented by the PW (or Part-Warehouse) relation. To establish this relationship in DB1 simply requires access to this relation. This is illustrated in Figure 9.

Figure 9 (a) represents the direction R1:E1-->E2. Recalling the rules of Section 5.4, traversal of this GDAG proceeds as follows.

1. Enter at the entry node. Since a vertical string does not exist for this node, horizontal traversal takes place.

2. Upon reaching the global entity node 1 for E1, vertical traversal must take place before horizontal traversal can continue. The * arc operation states that traversal continues to the first vertical attribute node. This node indicates that a global data element P#, with local name P#, must be obtained from a relation PW in database DB1. If a P# is obtained, the vertical string traversal is successful. Otherwise the GDAG traversal terminates. Since more than one P# might satisfy the graph, the vertical traversal must be repeated for each that does.

3. For each successful vertical traversal from the global entity node 1, horizontal traversal to the global entity node 2 for 2 for entity E2 takes place. This global entity node contains a vertical string which must be followed. The vertical string indicates a single global data element W#, with local data element name W#, must be obtained for the vertical traversal to

be a success. This W# is obtained from the PW relation accessed from the prior vertical traversal. This is indicated by the 1 qualifier preceding the PW in the upper right corner of the node.

4. Similarly, horizontal traversal continues to the global relationship node R for relationship R1. As with global entity node 2, the relationship information QTY is obtained from the PW relation accessed during the first vertical string traversal.

Figure 9 (b) represents the direction R1:E2-->E1. This GDAG is almost the same as Figure 9 (a), differing only in the order of usage of W# and P#.

Relationship R2 in DB1 is represented by the PSTR (or Part Structure) relation. To establish this relationship in DB1 simply requires access to this relation. This is illustrated in Figure 10.

Figure 10 (a) and (b) represent the directions R2:E1a-->E1c and R2:E1c-->E1a respectively. These GDAGs are almost the same as those in Figure 9, differing only in the usage of the relation PSTR compared to PW, the key field components P#A and P#C compared to P# and W#, and the relationship value QTY_USED compared to QTY.

CODASYL DB2

Figure 11 illustrates the Local GDAGs for the CODASYL database DB2. Since DB2 only implements relationship R1, only one set of local GDAGs are required for this database.

Whereas in DB1 the relationship R1 is contained in a single table PW, in DB2 this relationship is maintained as part of an owner (PART) and member (WH) set (INVENTORY). Because of this, the GDAGs for this relationship are more complicated than those of Figures

9 and 10. This is because database navigation using the set must take place.

Figure 11 (a) represents the direction R1:E1-->E2. To obtain the Warehouses which supply a Part requires establishing position on that Part as an owner and obtaining the Warehouse members associated to the Part via the INVENTORY set. This is what is stated in Figure 11 (a). A Part with some P# is obtained by traversal of the first vertical string. The next vertical string states that once position is achieved upon the Part in the first vertical string, follow some database access path (INVENTORY) to obtain a W# in WH. Once this WH is accessed, the QTY data element can be extracted to give a response to the query.

Figure 11 (b) represents the direction R1:E2-->E1. To obtain the Parts supplied by a Warehouse requires establishing position on that Warehouse as member and obtaining the Part owner associated to the Warehouse via the INVENTORY set. This is what is stated in Figure 11 (b). Since the CODASYL location mode of member WH is through the owner PART of the INVENTORY set, the first vertical string says to establish position on any PART. Then follow a database access path to find a WH with a particular W#. The next vertical string states that once position is achieved upon the Warehouse in the first vertical string, the PART and P# are automaticaly obtained via the positioning from the first vertical string. Similary, the QTY data element is immediately available once a WH is accessed from the first vertical string.

IMS/DB DB3

Figure 12 illustrates the Local GDAGs for the IMS/DB database DB3. Since DB3 only implements relationship R2, only one set of local GDAGs are required for this database.

DB3 is an IMS/DB hierarchical logical database. Because of the structure of the database, database navigation is required to establish the relationship R2.

Figure 12 (a) represents the direction R2:E1a-->E1c. In this context, the Assembly Part (E1a) is the root segment PART of DB3 and the Component Part (E1c) is the COMP_ASSEMB child segment of the root segment PART. Thus to obtain Component Part information of an Assembly Part is to establish position upon the root segment Part and to retrieve its COMP_ASSEMB children. This is what is stated in Figure 12 (a). A Part with some P# is obtained by traversal of the first vertical string. The next vertical string states that once position is achieved upon the Part in the first vertical string, follow some database access path (parent-child) to obtain a P# in COMP_ASSEMB. Once this COMP_ASSEMB is accessed, the QTY data element (with global name QTY_USED) can be extracted to give a response to the query.

Figure 12 (b) represents the direction R2:E1c-->E1a. In this context, the Component Part (E1c) is the root segment PART of DB3 and the Assembly Part (E1a) is the ASSEMB_COMP child segment of the root segment PART. Thus to obtain Assembly Part information of a Component Part is to establish position upon the root segment Part and to retrieve its ASSEMB_COMP children. This differs from Figure 12 (a) only in the use of the ASSEMB_COMP segment in comparison to the COMP_ASSEMB segment.

6.5 INTERDATABASE RELATIONSHIPS

Interdatabase relationships occur in two ways. One way is for the all or some (Vertical Partitioning [4]) attributes of a global entity to be maintained by one or more physical databases. For example, all three example physical databases maintain information about Parts. Some of the Part information may be replicated between physical databases. Some may be stored in only one physical database. The entire set of Parts is obtained as the union of Parts from each database. An interdatabase relationship of this type is termed an entity interdatabase relationship.

Interdatabase relationships may also occur due to the relationships between entities in the UGL ER view. For example, a Part at DB3 may be supplied by some Warehouse maintained at DB2 or DB1. A relationship of this type is termed a relationship interdatabase relationship.

Since GDAGs are for the purpose of specifying the access paths for the relationships contained in the ER view of the HDDB, access path specifications for entity interdatabase relationships are not included in this research. Only relationship interdatabase relationships are specified using GDAGs.

The interdatabase relationships due to relationships R1 and R2 that exist in the example HDDB are illustrated in Figures 13 and 14 respectively. Each of these require GDAGs to describe the access paths between databases to establish a interdatabase relationship. Some of these specifications are included below.

They key feature of these specifications is their similarity to the local GDAGs of Figures 9, 10, 11, and 12, differing only in complexity. This added complexity is required to establish an access path link between the physical databases. The similarity is of major significance in that it has the advantage that a global query analyzer may process the local and interdatabase GDAGs simultaneously, leaving problems of distribution to an access plan generator and global query coordinator.

6.6 INTERDATABASE RELATIONSHIP GDAG SPECIFICATIONS

The first set of interdatabase GDAGs are illustrated in Figure 15. These two GDAGs represent both directions of the interdatabase relationship R1 between the Parts of DB1 and the Warehouses of DB2.

Of particular interest in Figure 15 (a) is the vertical string attached to the horizontal node 2. Picking up the P# from the Part of DB1, a link is established with the P# in Part of DB2. This is indicated by the equivalence operator attaching the two constraint nodes. Once the Part in DB2 has been obtained, a database access path (the INVENTORY set) may be followed to access the Warehouse information. This Warehouse contains the Quantity On Hand of the Part as indicated in the vertical string of the horizontal node R.

Figure 15 (b) may be interpreted as follows. Some Part is obtained from DB2 which has a database access path (INVENTORY) for a particular Warehouse. The Warehouse is then used to establish a link with the PW table of DB1. This PW table contains the P# and QTY required to successfully establish the relationship.

The second set of interdatabase GDAGs are illustrated in Figure 16. These two GDAGs represent both directions of the interdatabase relationship R2 between the Assembly Parts of DB1 and the Component Parts of DB3. The two GDAGs bare a striking similarity to those of Figure 15. This is because the GDAG represents generalized access paths, independent of the type of database. What is required is the elementary operations and the order of operation needed to access the data.

7 CONCLUSION

The inherent query language and structural differences between different database management systems (DBMS) coupled with the difficulties of interdatabase relationships pose a major challenge in the automation of a Heterogeneous Distributed Database (HDDB). To combat this problem, a new database access path model, named the Generalized Database Access Graph (GDAG) is introduced. The GDAG is part of an integrated data dictionary, maintained as part of a HDDB Management System (HDDBMS), which guides HDDB query and data translation.

The GDAG is a two dimensional labeled directed graph which provides elementary operations at field level granularity which must be performed either by a local DBMS or a global query coordinator. This level of granularity is a necessity if existing application programs, utilizing a host DBMS Data Manipulation Language (DML) are to be given transparent access to the HDDB.

Two major capabilities feature the GDAG. Its high data independence allows relational, network, and hierarchical databases to be modeled using equivalent notation and structures. Second is the ability to model interdatabase relationships using the same data structure. This has the further advantage of allowing global query analysis to be performed independent of distribution. Distribution problems are the responsibiltiy of an access plan generator and global query coordinator.

REFERENCES

[1] Astrahan, M., et al., "Concepts Of A Data Independent Accessing Model," IBM Research Report RJ 1105, San Jose, California, October 1972.

[2] Cardenas, A.F., Nahouraii, E., and Brooks, L.O., "An Approach To Data Communication Between Different Generalized Data Base Management Systems," Systems For Large Databases, North Holland Publishing Company, 1976

[3] Cardenas, A.F. and Pirahesh, M.H., "The E-R Model In A Heterogeneous DBMS Network Architecture," in Chen, P. (ed.), Proc. Of The International Conference On The Entity-Relationship Approach To System Analysis And Design, North-Holland, Amsterdam, pp. 577-583, 1980.

[4] Cardenas, A.F., "Database Management Systems," 2nd Edition, Chapter 16, Allyn and Bacon, Inc., Boston, MA., 1985

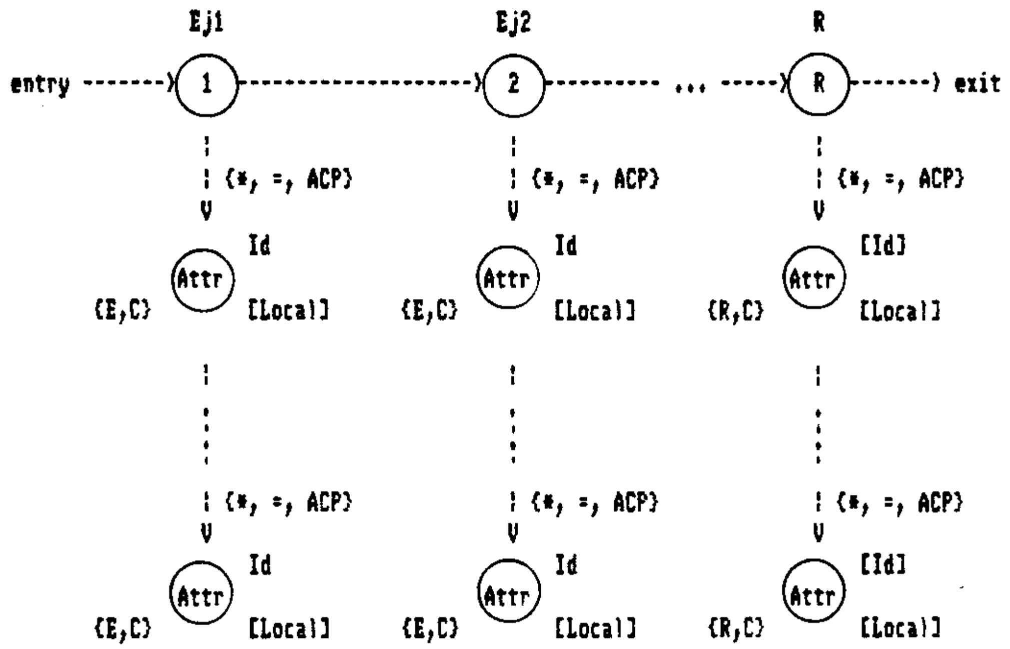
[5] Chen, P.P.S., "The Entity-Relationship Model - Towards A Unified View Of Data," Transactions On Database Systems, ACM, Vol. 1, No. 1, 1976

[6] Levin, M., "The DIAM Theory Of Algebraic Access Graphs," Sterling Systems Inc., 1980

[7] Pirahesh, M.H. and Cardenas, A.F., "Data Base Communication In A Heterogeneous Data Base Management System Network," Information Systems, Vol. 5, No. 1, pp. 55-79, 1980

```
                 Application Program

                        |  ^
                  query |  | results
         _______________V_ |_______________
        |                                  |
        |           Virtual Layer          |
        |__________________________________|
                        |
                        |  |
         _______________V_ |_______________
        |                                  |
        |      Unified Virtual Layer       |
        |__________________________________|
                        |
                        |  |
         _______________V_ |_______________
        |                                  |
        |       Unified Global Layer       |
        |__________________________________|
                        |
                        |  |
         _______________V_ |_______________
        |                                  |
        |        Unified Local Layer       |
        |__________________________________|
                        |
                        |  |
         _______________V_ |_______________
        |                                  |
        |           Local Layer            |
        |__________________________________|
                        |  ^
                        |  |
                        V  |

                   Local Database
```

Figure 1 : Layered architecture for the HDDBMS.



Legend :	[]	- optional entry
	{}	- choose one of a list of entries
	Attr	- unified global layer ER attribute name
	Id	- database identification
	Ej?	- { E1, E2, ... , En }
	Local	- local (physical) database data element name
	*	- continue logical access criterion
	=	- equal by value logical access criterion
	ACP	- database access path logical access criterion
	entry	- GDAG entry node
	exit	- GDAG exit node
	E	- entity node
	R	- relationship node
	C	- constraint node

Figure 2 : A generic Generalized Database Access Graph (GDAG).

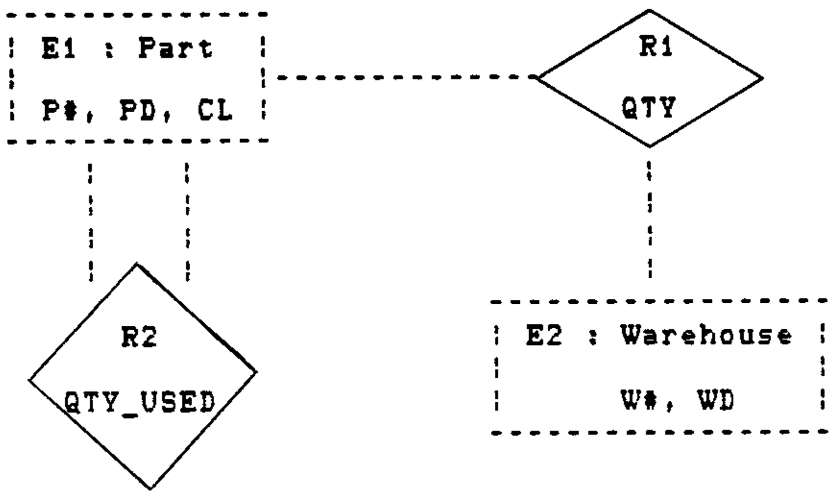


Figure 3 : Global ER diagram of the example HDDB.

```
CREATE TABLE PART { P# CHAR(5)     NON-NULL, /* PART NUMBER           */
                    PD CHAR(25)    NON-NULL, /* PART DESCRIPTION      */
                    CL CHAR(2)     NON-NULL  /* PART CLASSIFICATION   */
                  }
CREATE TABLE WH   { W# CHAR(5)     NON-NULL, /* WAREHOUSE NUMBER      */
                    WD CHAR(25)    NON-NULL  /* WAREHOUSE DESCRIPTION */
                  }
CREATE TABLE PW   { P#  CHAR(5)    NON-NULL, /* PART NUMBER           */
                    W#  CHAR(5)    NON-NULL, /* WAREHOUSE NUMBER      */
                    QTY DECIMAL(5) NON-NULL  /* QUANTITY ON HAND      */
                  }
CREATE TABLE PSTR { P#A CHAR(5)    NON-NULL, /* ASSEMBLY PART NUMBER  */
                    P#C CHAR(5)    NON-NULL, /* COMPONENT PART NUMBER */
                    QTY_USED                 /* QUANTITY OF THE       */
                        DECIMAL(5) NON-NULL  /* COMPONENT PART        */
                                             /* CONTAINED IN THE      */
                                             /* ASSEMBLY PART         */
                  }
```

Figure 4 : DB1 definition : a SQL/DS relational database.

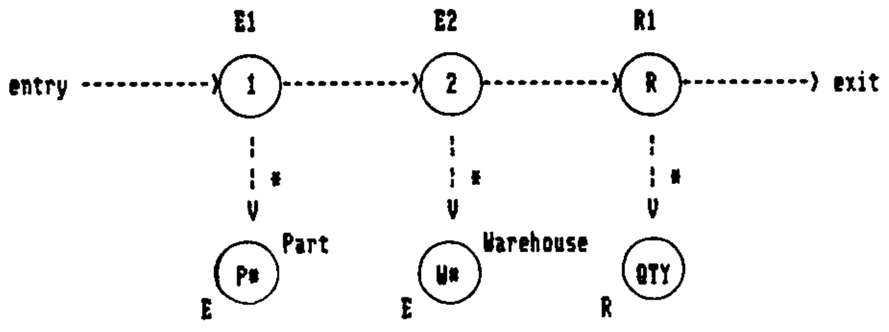
```
SCHEMA  NAME IS DB2.

AREA    NAME IS DB_AREA.

RECORD  NAME IS PART.
        LOCATION MODE IS CALC HASH-P#.
                 USING P# IN PART.
                 DUPLICATES NOT ALLOWED.

        WITHIN DB_AREA.

        02 P# TYPE IS CHAR  5.
        02 PD TYPE IS CHAR 25.
        02 CL TYPE IS CHAR  2.

RECORD  NAME IS WH.

        WITHIN DB_AREA.

        02 W# TYPE IS CHAR  5.
        02 WD TYPE IS CHAR 25.

SET     NAME IS INVENTORY.

        OWNER IS PART.

        MEMBER IS WH.

        MANDATORY AUTOMATIC.

        ASCENDING KEY IS W# IN WH.

        DUPLICATES ARE NOT ALLOWED.

        SET OCCURRENCE SELECTION IS THRU
            LOCATION MODE OF OWNER.
```

Figure 5 : DB2 definition : a CODASYL network database.

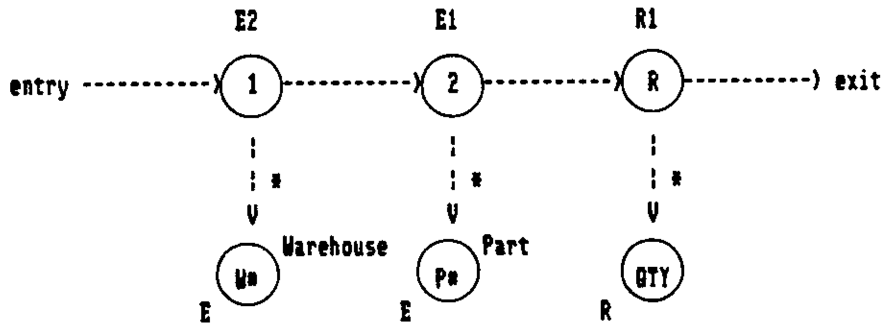
```
DBD      NAME=DB3,ACCESS=HISAM
DATASET  DD1=DEPTDD1,DEVICE=3380,OVFLW=DEPTOVF

SEGM     NAME=PART,BYTES=32
LCHILD   NAME=(COMP_ASSEMB,DB3),PAIR=ASSEMB_COMP
FIELD    NAME=(P#,SEQ),BYTES=5,START=1
FIELD    NAME=PD,BYTES=25,START=6
FIELD    NAME=CL,BYTES=2,START=31

SEGM     NAME=ASSEMB_COMP,BYTES=10,
         POINTER=(LPART,TWIN,LTWIN),
         PARENT=((PART),(PART,PHYSICAL,DB3))
FIELD    NAME=(P#,SEQ),BYTES=5,START=1
FIELD    NAME=QTY,BYTES=5,START=6

SEGM     NAME=COMP_ASSEMB,BYTES=10,POINTER=PAIRED,
         PARENT=PART,SOURCE=(ASSEMB_COMP,DB3)
FIELD    NAME=(P#,SEQ),BYTES=5,START=1
FIELD    NAME=QTY,BYTES=5,START=6
```

```
                    ---------------
                    !     PART    !
                    !             !
                    ! P#, PD, CL  !
                    ---------------
                           !
             ____________!____________
             !                       !
             !                       !
     ---------------         ---------------
     ! ASSEMB_COMP !         ! COMP_ASSEMB !
     !             !         !             !
     !   P#, QTY   !         !   P#, QTY   !
     ---------------         ---------------
```

Figure 6 : DB3 definition : an IMS/DB
hierarchical database.

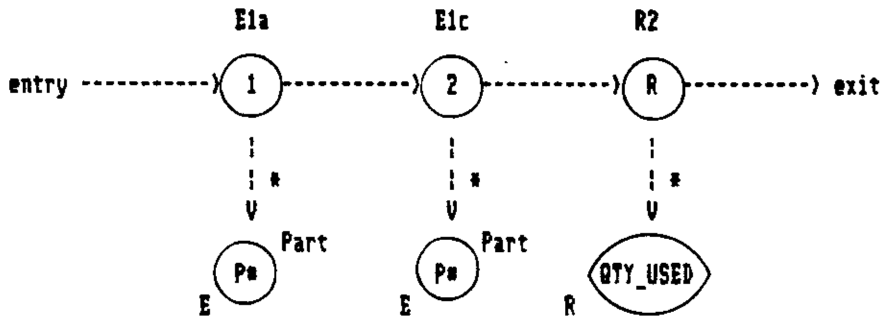


(a). R1 : E1-->E2

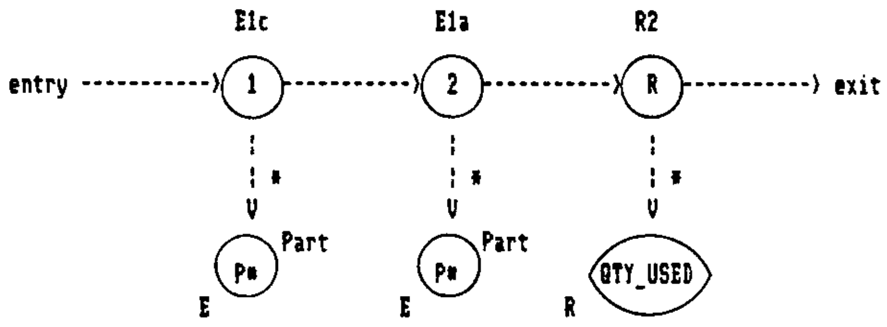


(b). R1 : E2-->E1

Figure 7 : R1 Global Relationship GDAGs.

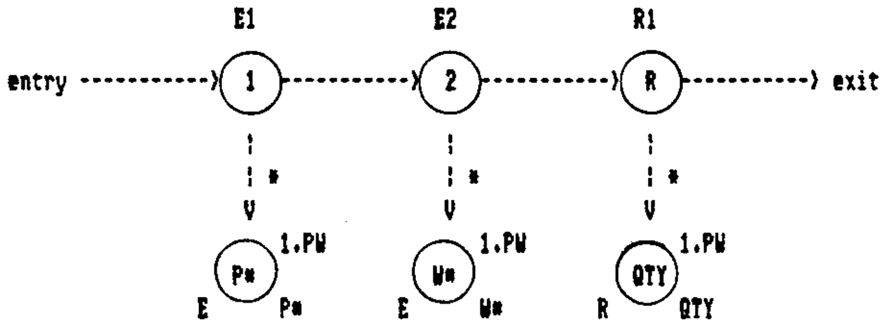


(a). R2 : E1a-->E1c

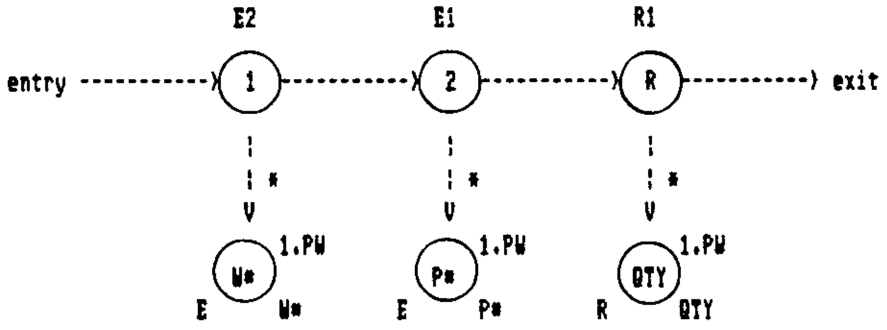


(b). R2 : E1c-->E1a

Figure 8 : R2 Global Relationship GDAGs.

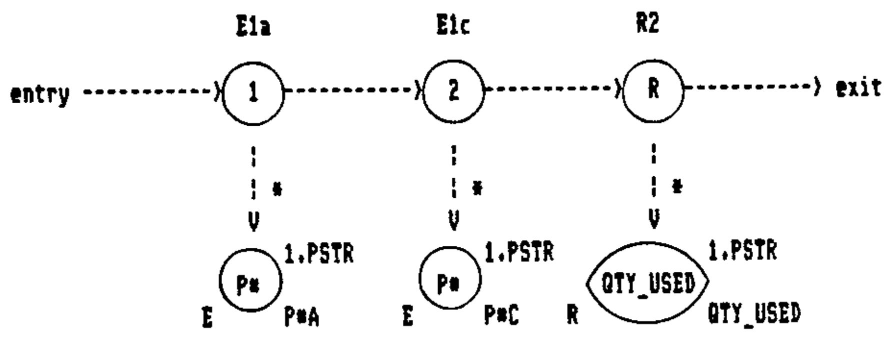


(a). DB1, R1 : E1-->E2

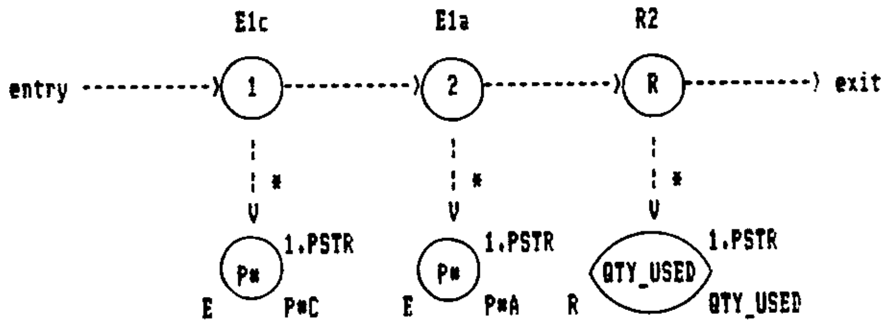


(b). DB1, R1 : E2-->E1

Figure 9 : Local DB1 R1 Local Relationship GDAGs.

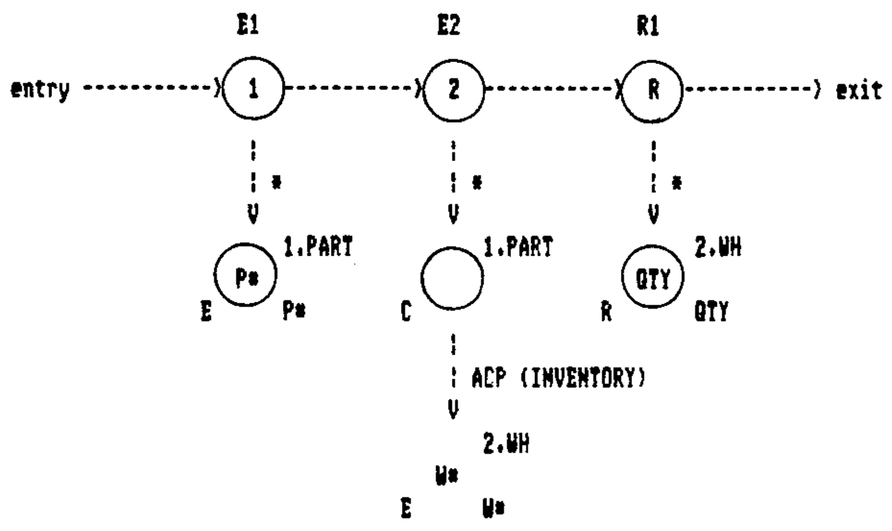


(a). DB1, R2 : E1a-->E1c

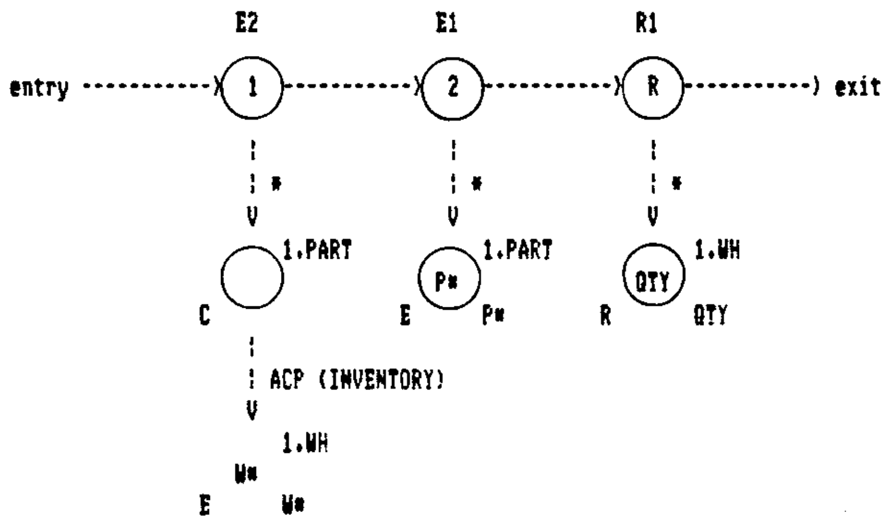


(b). DB1, R2 : E1c-->E1a

Figure 10: Local DB1 R2 Local Relationship GDAGs.

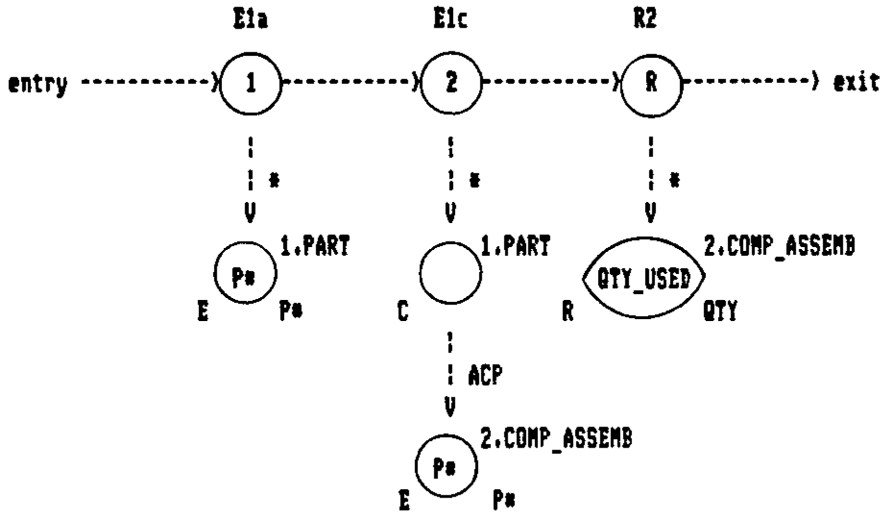


(a). DB2, R1 : E1-->E2

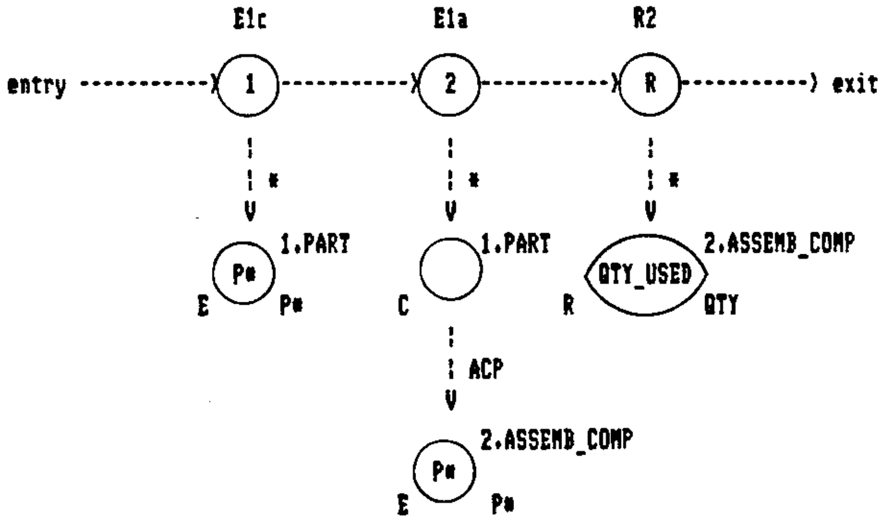


(b). DB2, R1 : E2-->E1

Figure 11 : Local DB2 R1 Local Relationship GDAGs.

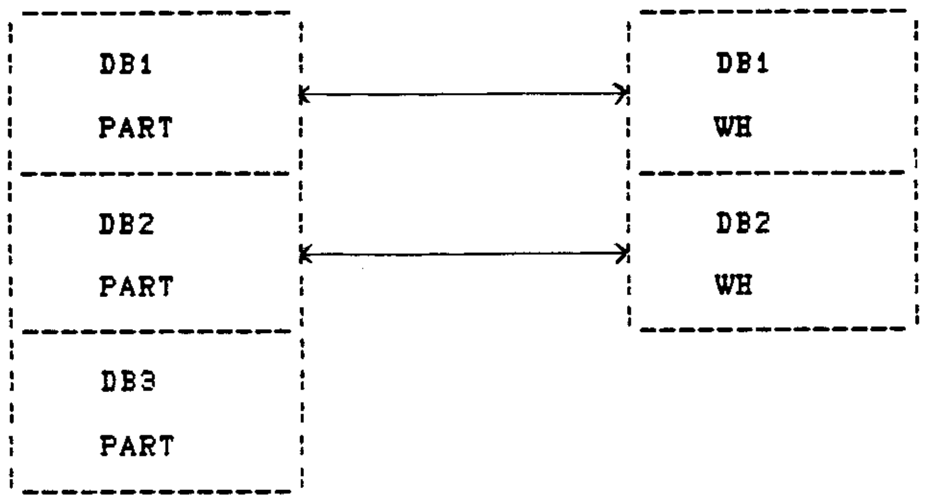


(a). DB3, R2 : E1a-->E1c

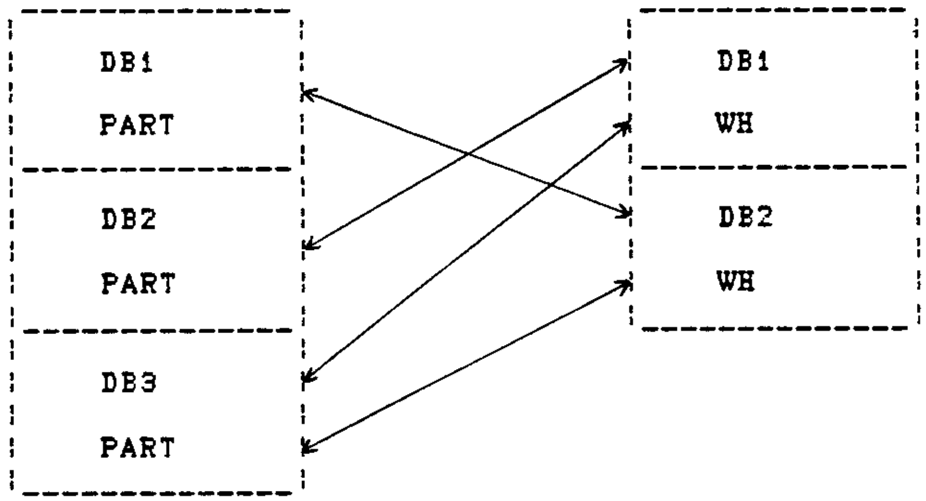


(b). DB3, R2 : E1c-->E1a

Figure 12 : Local DB3 R2 Local Relationship GDAGs.

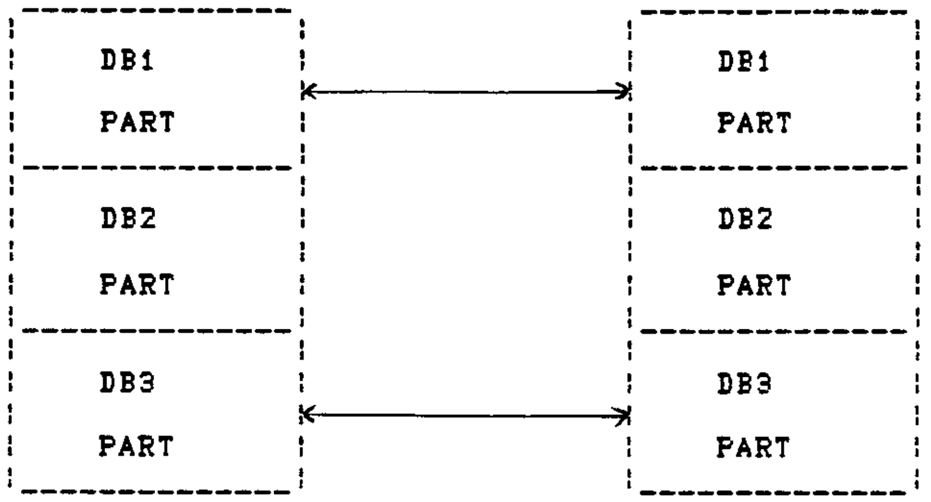


(a). Local database R1 relationships.

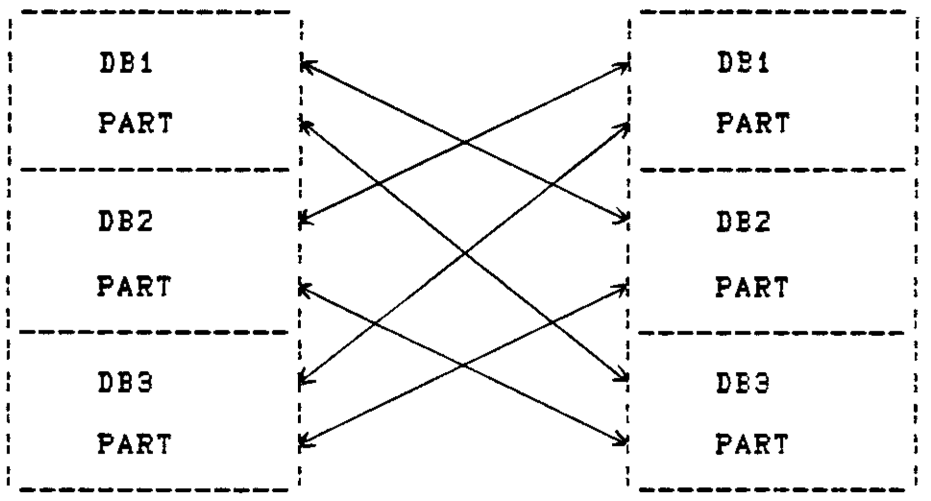


(b). Interdatabase R1 relationships.

Figure 13 : Local R1 relationship implementations compared to the interdatabase R1 relationships.

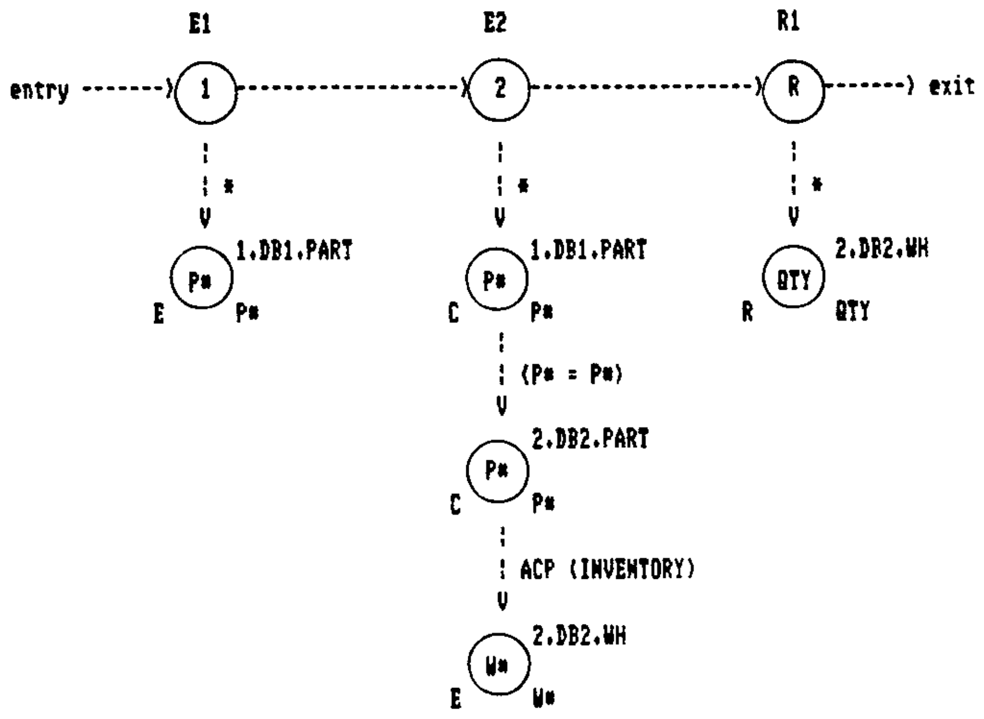


(a). Local database R2 relationships.



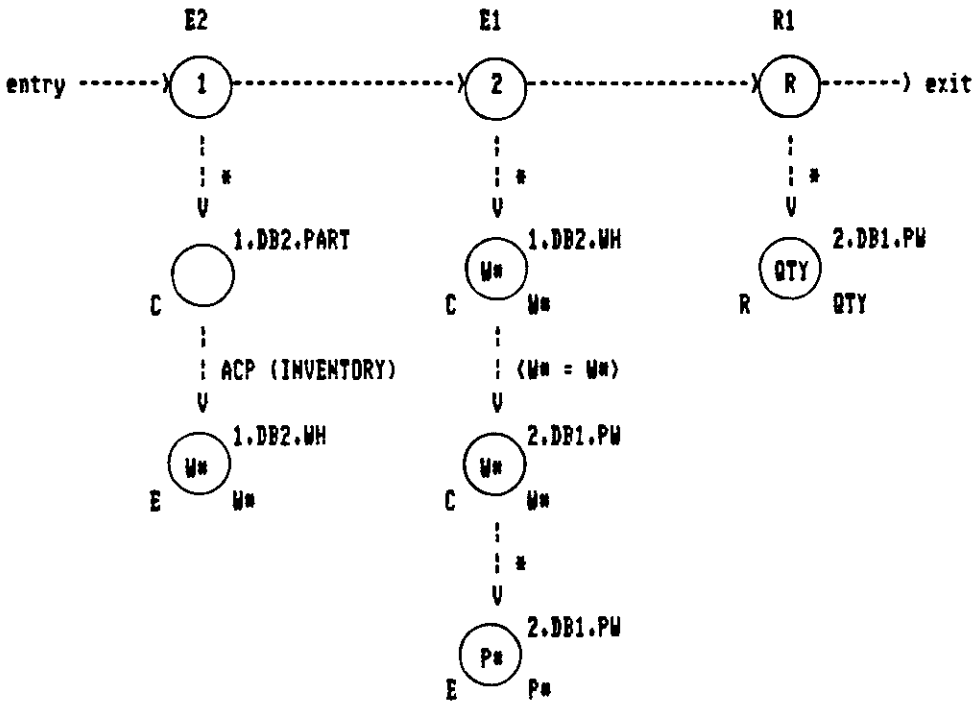
(b). Interdatabase R2 relationships.

Figure 14 : Local R2 relationship implementations compared to the interdatabase R2 relationships.



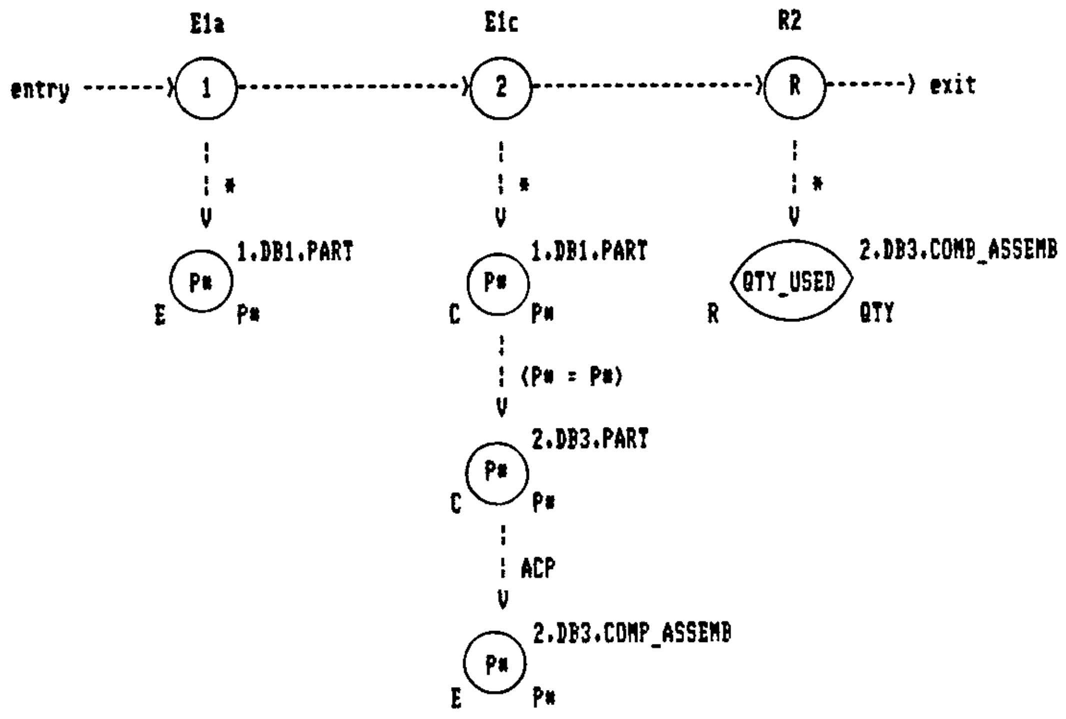
(a). R1 : E1-->E2 , DB1 (Part), DB2 (Warehouse).

Figure 15 : continued.



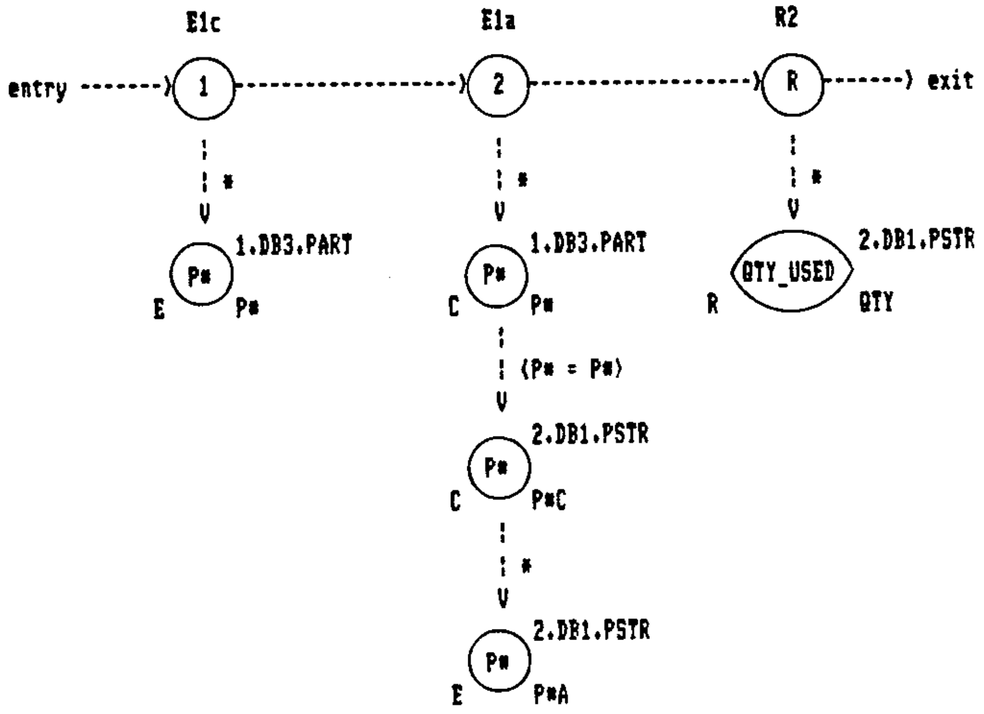
(b). R1 : E2-->E1 , DB1 (Part), DB2 (Warehouse).

Figure 15 : Interdatabase Relationship GDAGs.
R1 : DB1 (Part); DB2 (Warehouse)



(a). R2 : E1a-->E1c , DB1 (Assembly), DB3 (Component).

Figure 16 : continued.



(b). R2 : E1c-->E1a , DB1 (Assembly), DB3 (Component).

Figure 16 : Interdatabase Relationship GDACs.
R2 : DB1 (Assembly); DB3 (Component)