UNIBASE - An Integrated Access to Databases.

Z.Brzeziński J.Getta J.Rybnik W.Stępniewski
Institute for Scientific,Technical and Economic Information.
ul. Zurawia 3/5, - Warsaw, POLAND.

This Research is Supported by the Polish Computer Society
00-901 Warsaw, Poland, under Contract No. PTI-21/83/PIMB/W.

Abstract.
The concept of the integrated database system based on database logic is presented. The idea of an integrated database system may be used to deal with several database structures: hierarchical, network,relational, etc, applying one common data model. In the presented approach the relational model is established as a common view of different databases. The following parts of an integrated database system are described:
(1) a generalized schema of an integrated database schema and
(2) a formula language used for the definition of query, assertion, constraint, and transformation rules.

1. Introduction.

New applications of computer technology will be made possible by introducing the idea of an integrated database /Da1/,/U11/. By an integrated database we mean a data storing system residing on identical hardware under an identical Data Base Management System /software/ or in a different manner: a single schema /i.e., database description/ describing the entire database relative to which all accesses to the database are expressed. Such accesses are processed against a single /logical/ copy of the database.

Proceedings of the Tenth International Conference on Very Large Data Bases.

Unfortunately, in the real world many databases are not integrated. Often, data relavant to an enterprise is implemented by many independent databases, each with its own schema and own DBMS. In such cases, the database in addition to being nonitegrated is also distributed and heterogeneous. One of the basic reasons for such a situation is the application of various different data models.

The data model and the associated with it data sublanguage is the basic means by which data independence is achieved: the data model is the user's view of what is in the database, and the data sublanguage is the user's language for transferring data between the data model and his workspace. There are currently three favored approaches to database design /U11/: the relational approach /Co1/, the hierarchical approach /Da1/, and the network approach /Cd1/.

One of the principal problems in using distributed databases is the problem of integrated retrieval.In such databases, each independent database has its own schema expressed by its own data model, and it can be accessed only by its own retrieval language. Since in general different databases have different schemas and different retrieval languages, many difficulaties arise in formulating and implementing retrieval requests /called queries/ that require data from more than one database. These difficulaties include: resolving incompatibilities between the databases such as: differences of data types and conflicting schema names; resolving inconsistencies between copies of the same information stored in different databases; and transforming a query expressed in the user's language into a set of queries expressed in the many different languages supported by the different sites. Implementing such a query usually consumes months of programming time, making it a very

Singapore, August, 1984

expensive activity. Sometimes, the necessary effort is so great that implementing such a query is not feasible at all. This problem can be solved by constructing a software system binding several databases based on different data models. Its main goal is to present an illusion of an integrated database to users without requiring that the database be physically integrated Such a software system accomplishes this by allowing users to view the databases through a single common schema and by allowing them to access the data using a high level data manipulatin language. Queries posed in this language are entirely processed by the system as if the databases were homogeneous and integrated. An additional advantage of such an approach is the availability of application programs used in existing integrated databases.

2. The Model of Integration.

There are many approaches to the design of a system integrating the three most popular data models. Two of them are the most popular: the functional /Ba1/,/Bu1/,Sh1/, and the logical /Ja1/ approach. In deciding which approach to choose, we took into account the following design objectives:

(1) What kind of data model will be used on the integration level?
(2) What kind of a language will be used on that level?
(3) Does the system have to be specialized or general purpose?
(4) Is it to be only an interface or a system with functional extendability of integrated databases?
(5) How can we use the existing integrated database software?
(6) How to warrant flexibility of the system; we must be able to add additional attributes to the system as new research efferts come in?
(7) How to enable the transfer of the system between various hardware installation?

Having in mind the above consideration the relational data model was chosen for the design of the integration level of the database. The integration level is an additional level above the levels supporting the currently three favoured database models /i.e.,relational, hierarchical, and network/. In this way all of the advantages of the relational model /simplicity, universality, a good mathematical formalism, data independency, ease of operation on data items, data integrity, well defined functional dependencies, and the concept of schema normalization/ are kept without rendering the existing software invalid in the integrated database. Also all of the features of the relational model and the features of integrated databases are combined. An architecture of the designed system is depicted in figure 1.

The integrated database schema consists of three parts: User's Schema, Global Schema, and Transformation Rules.

The User's Schema implements the realational view of the database and allows the user to be independent of the complicated notions of both the Global Schema and Transformation Rules. It contains a Common Schema which can be accessed by all users and Private Schemas, accessed only by their owners /users who created them/. Therefore, the Global Schema consists of a number of Local Schemas and an Auxiliary Database Schema /relational/.Local Schemas and the Auxiliary Database Schema are defined in special formalism based on the concept of unnormalized /ONF/ relational schemas /Ul1/.

The Transformation Rules express how the basic elements of the User's Schema /i.e., relational schema/ can be obtained from the Global Schema.

It is assumed that each of the accessed DBMSs can be based on one of the three main data models /i.e., network, hierarchical, and relational/ and consist of a Local Host Schema and a Local Host DBMS. Every Local Host Schema must be mapped into one Local Schema. However, the transformation rules may join notions from many local schemas. Therefore, it is possible to create user's views that are based on many different databases.

The user's language is based on relational calculus /Pi1/. Due to the use of the relational data model any other language associated with that model can be utilized /e.g., SQL /As1/, DEDUCE 2 /Ca1/, QUEL /St1/, QBE /Z11/, or Relational Algebra /Co2//.

3. Design of the UNIBASE Architecture.

The proposed design of the system's architecture is called UNIBASE.

3.1. Basic Assumptions.

The applied relational data model of the UNIBASE system was extended with respect to the traditional Codd's model /Co1/ by the additional of derived Domains and Relations.We

call a Domain /Relation/ a derived Domain /Relation/ if it can be derived - using Definitions - from Domains/Relations/ existing in the database schema. Furtheremore, Consistency Constraints were added to the data model. They can be used for database logical integrity control. These additional features are embedded in the system's language.

Moreover, it is different in kind to the other integrated database systems for to the UNIBASE system an Auxiliary Database was inserted. The Auxiliary Database is based on the relational data model, too. This database is designed for storing primary relations containing data from the integrated databases. Such relations can appear in two forms:

(1) as effective relations - the relations are physically stored in the database, or
(2) as virtual relations - the rules for the creation of the relations are speciffied.

It should be noted that the access time of an effective relation is much shorter than of a virtual relation.

The Global Schema is nested between the User's level and the integrated databases level. It contains a number of local schemas which describe the integrated databases. Each of the local schemas has its own unique identifier. Apart from the global schema a set of the Transformation Rules exist. The Transformation Rules describe the translation process from the Global Schema notions to the User's Schema notions /relational view/, They allows the users to create relations from more than one of the integrated databases. The Global Schema was inserted for its mapping role from the main database models into one common data model /i.e., network, hierarchical or relational/.This level and its language is based on the DBL model /Ja1/.

The Auxiliary Database can be viewed by the users as one of the integrated databases. In such a case the Auxiliary Database /relational/ Schema should apper in the Global Schema as one of the local schemas. This means, that the user's relations can be interchanged among them through the integration of the Auxiliary Database.

Users can submit retrieval queries to the integrated databases. Each of the question is translated to the language of the integrated databases using DBL. The result of any query is a relation which is inserted into the Auxiliary Database in one of the above forms /i.e., effective or virtual/. Giving a status "to store" to the relation the user enables the insertion of it into the Auxiliary Database. If the relation's status is different then relation will be deleted in the latter part of the session.

In the UNIBASE system the integrated databases cannot be changed /updated/. Users can modify only effective relations contained in the Auxiliary Database, the virtual relations cannot be changed. It should be noted that update of the Auxiliary Database has no impact in the integrated databases contents.

3.2. The User's Language.

The User's Language can be used to pose questions to the integrated database through the UNIBASE system, to define derived domains and relations, and to formulate the consistency constraints. Furtheremore, in the extended user's language the DBA defines transformation rules. The proposed language is a modified version of a formula language defined for deductive management system /Ge1/. It belongs to the relation-domain calculus languages and is based on the formalism of first-order predicate logic. The user's language resembles slightly the DEDUCE 2 /Ca1/ language. However, its syntax makes it similar to natural languages. It contains the following improvements with respect to the DEDUCE 2 language: distributed relation names, calculus on domain and relation names,and join of relations names /phrases WHO, WHICH/.The language was also extended by the introduction of: numerical quantifiers, join of binary relations, transitive closure of binary relations, and elements from second-order predicate logic which are needed to define the transformation rules. The above-mentioned extensions of the user's language allow for a much greater expressiveness of the user's commands.

4. The Integrated Database Schema.

The Integrated Database Schema consists of the following parts:

(1) User's Schema:
- a specification of primary domain names,
- a specification of primary relations,
- a specification of names of derived domains,
- a specification of derived relations,
- definitions, and
- consistency constraints.

(2) Global Schema:
- a set of definitions of local schemas.

(3) Transformation Rules:
- a set of formulas expressing how the user's schema can be obtained from the global schema.

In each of the user's schemas we can select elements belonging to the Common User's Schema and the Private User's Schema.

The User's Common Schema is shared by all of the users and cannot be changed by them. It is built and maintained by the DBA.

The Private User's Schema cannot be shared by users and it contains domain and relation definitions inserted and accessed by a single user.

Below, all parts of the integrated database schema are discused.

4.1. The Specification of Names of Primary Domains.

The specifications of primary domain names contains domain names following the key-word PROMARY DOMAINS.

It was assumed that the following domain names are predefined in the UNIBASE system: INTEGER, REAL, STRING, and DATE.

4.2. The Specification of Primary Relations.

In this part of the User's Schema primary relations are defined. Each of the defined relations is specified as a string of domain names and distributed relation name elements. Relations can be defined if and only if their argument domains are defined.

4.3. The Specification of Derived Domains.

In this part the names of the derived domains are defined. Their specification is analogous to the definition of primary domains. Rules by which derived domains are derived from primary domains must be added to the User's Schema Definitions Area.

4.4. The Specification of Derived Relations.

The definition of derived relations is similar to the primary relations definition. All of its components must be defined earlier. As in the case of the derived domains here to the rules for deriving the derived relations from primary relations must be added to the User's Schema Definitions area.

4.5. Definitions.

Definitions /deriving rules/ have to be defined for all derived domains and relations. Each definition expressed in the user's language has its own unique name. All deriving rules are placed in the User's Schema Definitions area.

4.6. Consistency Constraints.

The Consistency Constraints area of the User's Schema consists of rules which have to be satisfied by the database contents. These rules ensure full logical security of the database during update and domain or relation derivation. Furtheremore, they make it possible to optimize the retrieval process by eliminating some parts of questions. Similar to Definitions each consistency contraints rule has its own unique name. All of the rules are defined in the user's language and placed in the User's Schema Constraints area.

The next two parts of the Integrated Database Schema /i.e., Global Schema and Transformation rules/ describe the logical model of the integrated database and transformation rules which describe how to "build" relations from data elements placed in the integrated databases.

4.7. The Database Global Schema.

The formalism used for Global Schema Definitions is based on the extended concept of ONF relational schemas /Ull/,/Jal/. The extensions consist of:

(1) The possibility of defining the same schema in many different manners and
(2) An integrated database identifier, which must be contained by each schema to associate every local schema with an appropriate local host schema.

All names used in the Global Schema Definitions must be unique.

4.8. The Transformation Rules.

The Transformation Rules area contains rules for transforming data items contained in the integrated databases. These rules are specified in the extended user's language. Every rule has its own unique name.

4.9. Schema Examples.

1. Let H be the hierarchical database schema defined as follows:

```
H = [COURSE[TITLE,COURSE#,DESCRIPT,
          PREREQ,OFFERING],
    PREREQ[COURSE#,TITLE],
    OFFERING[DATE,LOCATION,FORMAT,
    TEACHER, STUDENT],
```

TEACHER[NAME,EMP#],
STUDENT[NAME,EMP#, GRADE]].

The database schema H contains information about offered courses, teachers, students, dates, etc. We assume that our database is to contain the following information items:

(1) Who is the teacher of each student?
(2) What is the number associated with each person?
(3) Where and when are the courses offered?
(4) What lectures are offered at different locations?
(5) Which are the prerequisite courses for a given course?

The Integrated Database Schema corresponding to these assumptions is as follows:

```
SCHEMA
   PRIMARY DOMAINS EMP#,NAME,
        GRADE,DATE,LOCATION,FORMAT,
        TITLE,COURSE ,DESCRIPT
   PRIMARY RELATIONS NAME IS-NAME-
        OF-TEACHER EMP#,
        NAME IS-NAME-OF EMP#
        RANKED GRANDE,
        TITLE IS-TITLE-OF-COURSE-
        -NO COURSE,
        EMP# IS-TEACHER-OF EMP#
        AT-LOCATION LOCATION
        FROM DATE WITH FORMAT,
        COURSE# TITLED TITLE
        DESCRIBED-BY DESCRIPT
        IS-OFFERED-AT LOCATION
        FROM DATE,
        COURSE# TITLED TITLE IS-
        -NEEDED-TO-PASS-THE-
        COURSE COURSE#
   DERIVED DOMAINS MATHEMATICS,
        PHYSICS,EUROPE
   DERIVED RELATIONS EMP# STUDIES-
        -AT LOCATION,
        EMP# IS-STUDENT-OF EMP#
   DEFINITIONS MATH: x:MATHEMATICS=
        = x: TITLE AND x: 'algebra'
        OR 'topology' OR 'geometry' ;

        PHYS: x:PHYSICS=x:TITLE AND
        x: ( 'mechanics' OR
        'relativity theory'
        OR 'electricity' ) ;
        EUROPE: x:EUROPE=x:LOCATION
        AND x: ( 'madrit' OR 'london' OR
        'paris ');
        STUDY: x STUDIES-AT y=EXIST
        z:EMP ,d:DATE,
        f:FORMAT,l:LOCATION
        (z IS--TEACHER-OF x
        AT-LOCATION 1 FROM d WITH f)
        TEACHES: x IS-STUDENT-OF
        y= EXIST d:DATE,
        l:LOCATION,f:FORMAT
        (y IS-TEACHER-OF x
        AT-LOCATION 1
        FROM d WITH f );
   CONSTRAINTS ANTISYMMETRY:
        FOR ALL x,y (x IS-STUDENT-
        -OF y IMPLY y NOT IS-
        -STUDENT-OF x) ;
        NONREFLEXIVITY: FOR ALL x:
        EMP#(x NOT IS-STUDENT-OF x;
   GLOBAL SCHEMA
        COURSE[COURSE#, TITLE,
        DESCRIPT,PREREQ,OFFERING]H
        PREREQ[TITLE, COURSE#]H,
        OFFERING[DATE, LOCATION,
        FORMAT,TEACHER,STUDENT]H,
        TEACHER[NAME,EMP#]H,
        STUDENT[NAME,EMP#,
        GRADE]H
   TRANSFORMATIONS
        T1 : x IS-NAME-OF-
        -TEACHER y = TEACHER[x,y];
        T2 : x IS-NAME-OF y RANKED
        z = STUDENT [ x,y,z];
        T3: x IS-TITLE-OF-COURSE-
        -NO y = PREREQ [x,y];
        T4: x IS-TEACHER-OF y AT-
        LOCATION 1 FROM d WITH f=
        = EXIST t: TEACHER, s: STU-
        DENT, g:GRADE,ns:NAME, nt:
        NAME (OFFERING [d,1,f,t,s]
        t[nt,x] AND s[ns,y,g]);
        T5: x TITLED y DESCRIBED-
        -BY z IS-OFFERED-AT 1 FROM
        d = EXIST off: OFFERING,
        p: PREREQ (EXIST f,t,s
        (off[d,1,f,t,s] AND COURSE
        [x,y,z,p,off]));
        T6: x TITLED t IS-NEEDED-
        -TO-PASS-THE-COURSE y =
        =EXIST preq:PREREQ, off:
        OFFERING, tit,d (preq[t,x]
        AND COURSE[y, tit,d,preq,off]);
```

2. Presented below is an integrated database schema corresponding to a simple network database - N. The network schema is defined in the Global Schema part of the Integrated Database Schema.

```
SCHEMA
   PRIMARY DOMAINS CHIEF# ,EMP#
   PRIMARY RELATIONS CHIEF IS-
```

-MANAGER-OF EMP#
CONSTRAINTS ASYM: FOR ALL x,y (x IS-MANAGER-OF y IMPLY y NOT IS-MANAGER-OF x)
GLOBAL SCHEMA MANAGER[C#, E#]N; MANAGER [C#,MANAGER]N
TRANSFORMATIONS
MNG1: x IS-MANAGER-OF y = MANAGER [x,y] ;
MNG2: x IS-MANAGER-OF y = EXIST c: C#, v:. IS-MANAGER--OF. (MANAGER[x,v] AND v[c,y]);

The sign "." determines the position of arguments of the relation.

5. Access to Integrated Databases.

5.1. Query Processing Subsystem.

A database language permits users to pose queries to the UNIBASE system. The main command in database language is find, which expresses the natural guery - "find all elements satisfing a formula".

A submitted query is processed in the following way. At first, the syntactic analysis of the formula is performed. The validity and type (primary or derived) of every domain and relation name occuring in the formula are established. During the phase of the syntactic analysis the formula is converted to the binary tree form. Then, all domains and relations of the derived type are substituted by the adequate formulas from the Definitions part of the Integrated Database Schema. The trace of the performed substitutions is stored on the stack and may be utilized in the process of answering the query of the type WHY. Next, the query formula may be checked the logical integrity constraints. Sometimes, the answer to the query may be found without the retrieval in the database, but only by applying the consistency constraints which are valid in this database. In order to perform this phase, the formula is converted to the clause form and the well known methods adapted from automated theorem proving area (resolution principle) are used /Ca2/. Since this phase is rather time-consuming, its execution is optional. Then, using the table of domains and relations names appearing in the formula, the adequate transformation rules from Transformation area are selected. From each transformation rule one job is created. Such jobs are then submitted to the respective integrated databases, where they are executed. Every job contains the retrieval program generated on the ground of the transformation rule and global schema. This program is written in Pascal-like language the Generalized Data Manipulation Language. The basic constructions of this language will be described in the next section. It is assumed that every Database Management System contains the interpreter of the Generalized Data Manipulation Language. The results from all jobs are sent to the host installation where they are compressed /repeated tuples are removed/ and merged into required relations. These relations are then stored in the Auxiliary Database of a particular installation and are used for retrieval. They are automatically removed at the end of the user session. However, the user can change status of the created relations and keep them in the Auxiliary Database for any period of time. The process of the user's query processing is schematically presented in figure 2.

5.2. The Generalized Data Manipulation Language.

The Generalized Data Manipulation Language is intended to be used in the environment of relational, network and hierarchical databases. It is assumed that the language operates on the tabular /unnormalized relation/ form of data. The language possesses standard Pascal-like control structures such as while .. do .., if .. then .. else, case .. do, etc. Moreover, there exist a number of standard functions and procedures intended to operate on unnormalized relations. It is assumed that these functions are supplied by the Administrator of the Integrated Database. It enables physical access to database files so it must be separately created for every integrated database. The set of basic data access functions is presented below:

(1) function SYSTEM-TABLE- enables access to system directory which contains addresses of all tables stored in the database,
(2) function SELECT-SUB-TABLE (TUPLE, TABLE) extracts the table which is contained in a given tuple,
(3) function END-OF-TABLE (TABLE) - boolean function which tests if the end of the table is encountered,
(4) function GET-NEXT-TUPLE (TABLE) - enables access to the next tuple in the table,
(5) procedure SEND(TUPLE,DESTINATION)

- procedure realizes transmission of a tuple to given integrated database installation.

Moreover, it is assumed that operator " * " may be used to concatenate two tables.

Example 5.2.1.

Let us assume the following transformation rule:

```
x SUPPLIES y TO z = EXIST p: PARTS,
                    prj:PROJECTS,
                    m: MANAGER
                   (SUPPLIER[x,p] AND
                    p[y, prj] AND
                    prj[z,m1]);
```

The Generalized Data Manipulation Language program corresponding to the above rule is presented below:

```
PROGRAM SUPPLIES-TO;
 VAR T,SUPPLIER,PARTS,PROJECTS:
     TABLE; NAME,PART# ,PROJECT#:
     ATTRIBUTE; SUPPLIER-TUPLE,
     PARTS-TUPLE,
       PROJECTS-TUPLE : TUPLE;
 BEGIN
  T := SYSTEM-TABLE;
  SUPPLIER := SELECT-SUB-TABLE(T,
                   'SUPPLIER');
  WHILE NOT END-OF-TABLE(SUPPLIER)
   DO
    BEGIN
     SUPPLIER-TUPLE := GET-NEXT-
                       TUPLE(
                       SUPPLIER) ;
     NAME := SELECT-SUB-TABLE(
               SUPPLIER,
               SUPPLIER-TUPLE) ;
     PARTS := SELECT-SUB-TABLE(
                     SUPPLIER-
                    -TUPLE, 'PARTS');
     WHILE NOT END-OF-TABLE(PARTS)
     DO
      BEGIN
       PARTS-TUPLE := GET-NEXT-TU-
                      PLE(PARTS);
       PART := SELECT-SUB-TABLE(
               PARTS-TUPLE,PART#) ;
       PROJECTS := SELECT-SUB-
                   -TABLE(PARTS-
                   -TUPLE
                   'PROJECTS') ;
       WHILE NOT END-OF-TABLE(
        PROJECTS) DO
        BEGIN
         PROJECTS-TUPLE := GET-NEXT-
                           TUPLE PROJECTS);
         PROJECT := SELECT-SUB-
                    TABLE(
                    PROJECTS-TUPLE,
                    'PROJECT#');
         RESULT-TUPLE := NAME * PART#
                         * PROJECT ;
         SEND(RESULT-TUPLE,DESTINA-
          TION)
        END
      END
    END
  END.
```

SUMMARY.

This paper establishes the fundamental ideas and properties of the UNIBASE system. Details of the systems architecture to be implemented in the initial breadboard vision are also described. Although, additional research is required to fill in the details of optimization and incompatible data handling. The presented architecture already contains several innovative ideas in integrating distributed heterogeneous databases. These include:

(1) The idea of using database logic to describe the logical database schema of integrated databases and
(2) The idea of using transformation rules to describe a way to "build" relations from data elements placed in the integrated databases.

As the next move we will intend to investigate the methods of updating the integrated databases and the Auxiliary Database without violation of the integrity constraints of each of the databases considered.

The main future goal of the design is to implement the UNIBASE system on distributed hardware - a microprocessors net. An attempt will be made to describe this idea and our implementation experience with the UNIBASE system in the next paper. Moreover, the UNIBASE as a Distributed Data Base Management System in local computer networks and its application areas will be presented elsewhere.

ACKNOWLEDGEMENTS.

The author wish to thank K.Turek who helped so much in the development of this paper. Sincere thanks are also given to A.Gustowski, J.Lizoń, D.Pawlak, S.Romański, and H.Rybiński for their comments and suggestions.

REFERENCES.

/As1/ Astrahan M.M. Chamberlin D.D.

Implementation of a Structured English Query Language.
CACM Vol.18, No.10, 1975.

/As2/ Astrahan M.M., et al.,
System R: A Relational DBM System.
IEEE Computer Vol. 13, No.5, 1979.

/Ba1/ Backus J.,
Can Programming be Liberated from the von Neumann Style? A Functional Style and its Algebra of Programs.
CACM Vol. 21, No.8, 1978.

/Bl1/ Blasgen M.W., et.al.,
System R: An Architectural Overview.
IBM Systems Journal Vol. 20, No.1, 1981.

/Bu1/ Buneman P., Frankel R.E.,
FQL - A Functional Query Logic.
Proc. ACM SIGMOD Conf.,Boston Mass. 1979.

/Ca1/ Chang C.L.,
DEDUCE 2: Further Investigations of Deduction in Relational Data Bases.
Logic and Data Bases ed. H.Gallaire, J.Minker, 1978.

/Ca2/ Chang C.L.,
On Evaluation of Queries Containing Derived Relations in a Relational Data Base.
Advances in Data Bases Theory ed. H.Gallaire, J.Minker, J.Nicolas, Plenum Press 1981.

/Cd1/ CODASYL Systems Committe,
Introduction to Feature Analysis of Generalized Data Base Management Systems.
CACM Vol. 14, No.5, 1971.

/Ch1/ Chamberlin D.D., et al.,
SEQUEL 2: A Unified Approach to Data Definition, Manipulation an Control.IBM Journal of Res. and Dev. Vol.20, No.6,1976.

/Ch2/ Chamberlin D.D., Gilbert A.M., Yost R.A.,
A History of System R and SQL/DATA System.
Proc.Very Large Data Bases, Cannes 1981.

/Co1/ Codd E.F.,
A Relational Model of Data for Large Shared Data Banks.
CACM Vol. 13, No.16,1970.

/Co2/ Codd E.F.,
Relational Completness of Data Base Sublanguage.
Data Base Systems ed. R.Rustin, 1971.

/da1/ Date C.J.,
An Introduction to Database Systems.
Addison-Wesley 1975.

/Ge1/ Getta J., Rybiński H.,
Deductively Augmented Database Management System.
Information Systems /to appear/.

/Ja1/ Jacobs B.E.,
On Database Logic.
Journal of the ACM Vol.29, No 2,1982.

/Ma1/ Maier D.,
Theory of Relational Databases.
Addison-Wesley 1983.

/Mu1/ MULTIBASE - A Research Program in Heterogeneous Distributed DBMS Technology.
Technical Report of the Defense Advanced Research Project Agency of the Department of Defense and the Naval Electronic System Command No. N00039-80-C-0402.

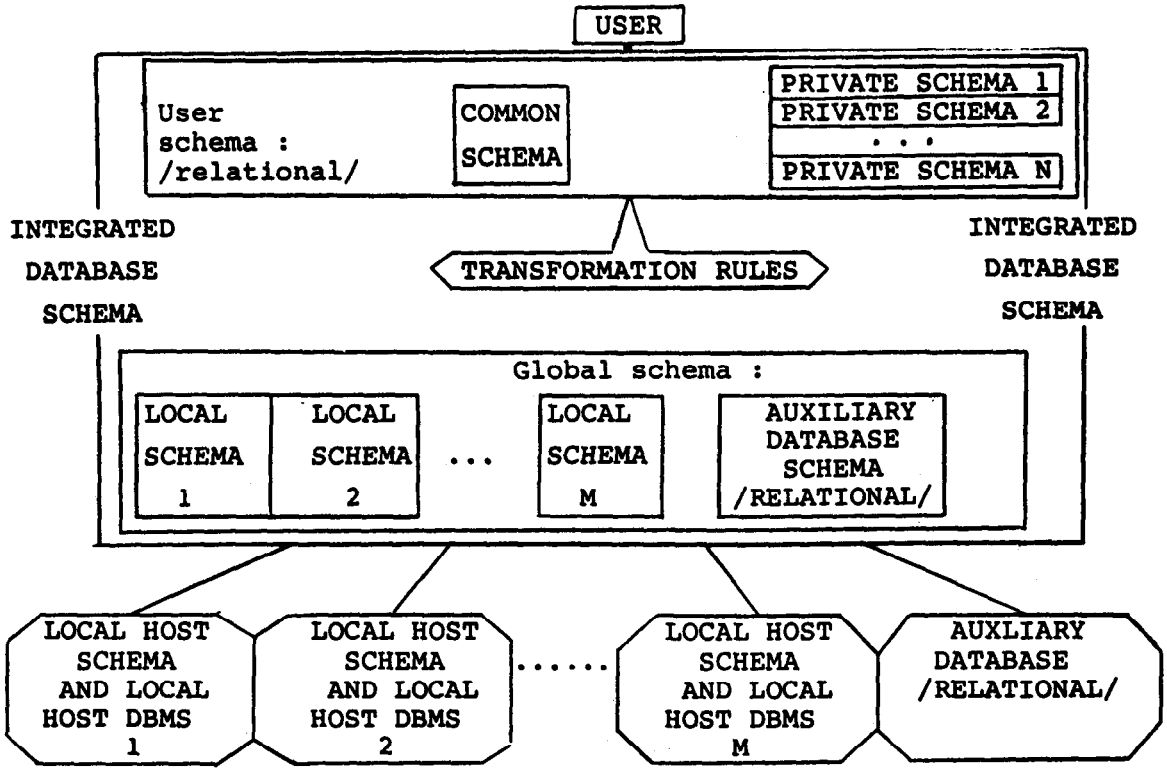
/Pi1/ Pirottle A.,
High Level Data Base Query Language.
Logic and Data Bases ed. H.Gallaire, J.Minker, 1978.

/Sh1/ Shipman D.W.,
The Functional Data Model and the Data Language DAPLEX.
ACM Trans. on Database System Vol.6, No.1, 1981.

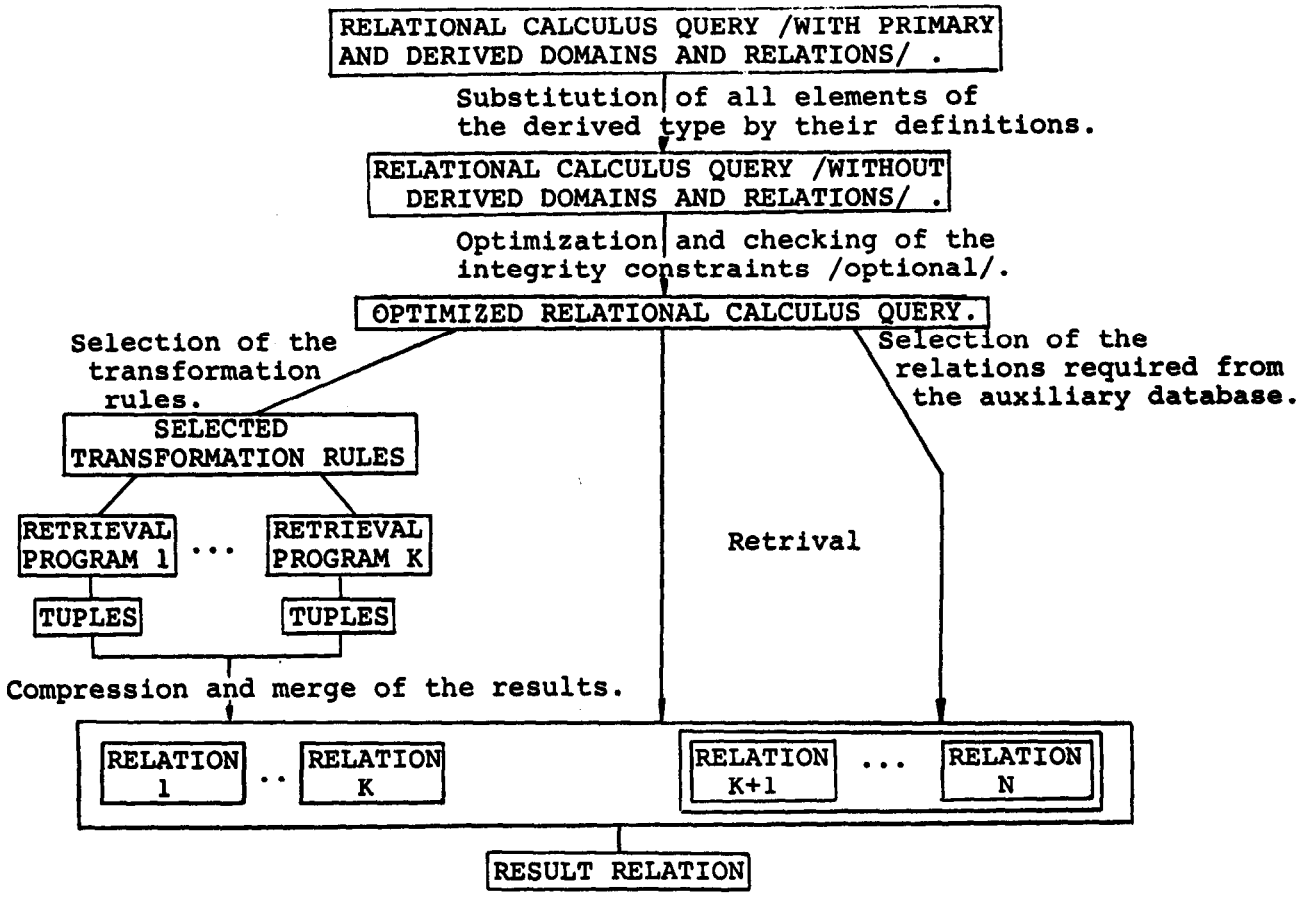
/St1/ Stonebraker M., Wong E., Kreps P.,
The Design and Implementation of INGRES.
ACM Trans. on Database System Vol.3, No.1, 1978.

/Ul1/ Ullman J.D.,
Principles of Database Systems.
Computer Science Press 1970.

/Zl1/ Zloof M.M.,
Query By Example: A Data Base Language.
IBM Systems Journal Vol.16, No.4, 1977.


USER
User schema : /relational/
COMMON SCHEMA
PRIVATE SCHEMA 1
PRIVATE SCHEMA 2
. . .
PRIVATE SCHEMA N
INTEGRATED DATABASE SCHEMA
TRANSFORMATION RULES
INTEGRATED DATABASE SCHEMA
Global schema :
LOCAL SCHEMA 1
LOCAL SCHEMA 2
...
LOCAL SCHEMA M
AUXILIARY DATABASE SCHEMA /RELATIONAL/
LOCAL HOST SCHEMA AND LOCAL HOST DBMS 1
LOCAL HOST SCHEMA AND LOCAL HOST DBMS 2
.....
LOCAL HOST SCHEMA AND LOCAL HOST DBMS M
AUXLIARY DATABASE /RELATIONAL/

/Fig.1/


RELATIONAL CALCULUS QUERY /WITH PRIMARY AND DERIVED DOMAINS AND RELATIONS/ .
Substitution of all elements of the derived type by their definitions.
RELATIONAL CALCULUS QUERY /WITHOUT DERIVED DOMAINS AND RELATIONS/ .
Optimization and checking of the integrity constraints /optional/.
OPTIMIZED RELATIONAL CALCULUS QUERY.
Selection of the transformation rules.
Selection of the relations required from the auxiliary database.
SELECTED TRANSFORMATION RULES
RETRIEVAL PROGRAM 1
...
RETRIEVAL PROGRAM K
Retrival
TUPLES
TUPLES
Compression and merge of the results.
RELATION 1
..
RELATION K
RELATION K+1
...
RELATION N
RESULT RELATION

/Fig.2/