Cluster Algebra: A Query Language for Heterogeneous Databases

Bharat Bhasker, Csaba J. Egyhazy, and Konstantinos P. Triantis

TR 92-57

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

December 21, 1992

Cluster Algebra: A Query Language for Heterogeneous Databases

Authors & Affiliations

Bharat Bhasker
Hughes STX Corporation
Lanham, MD 20706
Tel: (301) 286-4110

Csaba J. Egyhazy
Department of Computer Science
VPI & SU
Northern Virginia Graduate Center
2990 Telestar Court
Falls Church, VA 22042
Tel: (703) 698-6021

Konstantinos P. Triantis
Department of Industrial & Systems Engineering
VPI & SU
Northern Virginia Graduate Center
2990 Telestar Court
Falls Church, VA 22042
Tel: (703) 698-6086

Abstract

This report describes a query language based on algebra for heterogeneous databases. The database logic[7] is used as an uniform framework for studying the heterogeneous databases. The data model based on the database logic is referred to as *cluster data model* in this report. Generalized Structured Query Language (GSQL) is used for expressing ad-hoc queries over the relational, hierarchical and network database uniformly. For the purpose of query optimization, a query language that can express the primitive heterogeneous database operations is required. This report describes such a query language for the clusters (i.e. heterogeneous databases). The cluster algebra consists of (a) Generalized relational operations such as selection, union, intersection, difference, semi-join, rename and cross-product; (b) Modified relational operations such as normal projection and normal join; and (c) New operations such as normalize, embed and unembed.

Table of Contents

1 Introduction

The design and development of heterogeneous distributed database management systems (HDDBMSs) [5] is an ongoing research and application problem. Within the context of this problem, the issues of the heterogeneity of the underlying DBMSs and of efficiently accessing the distributed data have been important [4]. Many theoretical solutions have been proposed and different prototype HDDBMSs provide varying implementation approaches for these issues [12]. In this report a low-level query language based on algebra, called cluster algebra is described. The cluster algebra generalizes the relational algebra from relational to the heterogeneous case. It also introduces several new operators similar to [10] operators for nested relations. The new operators introduced here have the scope of relational hierarchical as well as network databases.

In the past couple of years, various approaches to express non-relational models in terms of relational constructs have been proposed [1,2,4,5,6,7,8,9,10,11]. Most of these approaches are based on Mackinouchi's [9] proposal to abandon the first normal form condition placed on relational databases. However, most of these approaches address only hierarchical data. Few general frameworks deal with hierarchical and network data models. These frameworks are: the format model [6], the logical data model [8] and database logic [7]. Database logic is an extension of first order logic with the capability to deal with hierarchical and network databases.

2 The Cluster Data Model

The history of databases is the history of data models. Every decade or so a new data model becomes prominent. This has resulted in an abundance of data models, producing the problems germane to a system of heterogeneous databases. The approach presented in this report is the treatment of multiple data models in an uniform way. In attempting to build an architecture to implement such uniformity, the initial step is the formulation of a data model that embraces all the underlying data models.

This paper reports on the implementation of such a data model, herein called the cluster data model, that embraces the following three underlying data models: network, hierarchical and relational.

The concept of cluster is based on the database logic [7]. Database logic is a generalization of the relational model and unifies the relational, hierarchical and network database schemas.

A database schema S is a finite collection of rules of the form $R_j = (R_{j1}, R_{j2}, \ldots, R_{jk})$. The objects R_j and R_{ji}'s are called names. Higher order names appear on the left hand side of the rules. The names, which appear only on the right hand side of the rules are called zero order names. A name R_j that appears only on the left hand side of rules, is referred to as an *external name,* otherwise, it is referred to as an *internal name*. For S to be a database schema, it must satisfy the following conditions:

i. No two rules can have same higher order name on the left hand side.

ii. Zero order names and higher order names are distinct.

iii. Names on the right hand side of same rule are unique.

Special classes of schemata can be defined by placing restrictions on the set S.

A relational schema RS is a collection of rules $RS = \{R_j = (R_{j1}, R_{j2}, \ldots, R_{jk})\}$ where, R_{ji} for $i = 0, 1, 2, \ldots k$ are all zero order names and RS is a heterogeneous database schema.

A hierarchical schema HS is a collection of rules $HS = \{\ldots, R_j = (R_{j1}, R_{j2}, \ldots, R_{jk}), \ldots\}$ where :

i. For all j, $R_j = (\ldots, R_j, \ldots) \notin HS$

ii. No higher order name appears on the right hand side of two different rules.

iii. For all valid subcluster access paths[1] $(R_0, R_1,..,R_i, R_{i+1},..., R_r)$ of the schema, a higher order name R_j does not appear more than once.

iv. H is a heterogeneous database schema.

A network schema N is a collection of rules $N = \{...,R_j = (R_{j1}, R_{j2},......, R_{jk}),......\}$ where:

i. For all j, $R_j = (....,R_j,....) \notin N$

ii. For all valid subcluster access paths[2] $(R_0, R_1,..,R_i, R_{i+1},..., R_r)$ of the schema, a higher order name R_j does not appear more than once.

iii. N is a heterogeneous database schema.

The access path for a database schema is defined as follows:

Subcluster Access Path Definition:

Let $S = \{...,R_j = (R_{j1},R_{j2},......,R_{jk}),......\}$ be a collection of rules. A *Subcluster Access Path* in S is a list of the form:

$(R_0, R_1,..,R_i, R_{i+1},..., R_r)$

where, for i = 0,1,.......,r, the following hold:

i. R_o is an external name;

---

1 The concept of a subcluster access path is defined subsequently in this section.

2 The concept of a subcluster access path is defined subsequently in this section.

ii. R_i is the higher order name;

iii. R_{i+1} appears on the right hand side of rule R_i.

The cluster is an instantiation of the cluster schema. The cluster schema defines databases in terms of one or more interrelated tables. This means that an attribute can have tables as its domain. An attribute with a table domain is considered to have a sub-table as its value. The sub-table instances may vary through time. However, they must be of fixed type and defined somewhere in the cluster. This table and sub-table relationship allows the representation of a link for the hierarchical model. By including a sub-table in two or more tables, one can represent the network model [3].

The Cluster

The database structure corresponding to the relational, hierarchical and network database schema is referred to as cluster. In this research, a cluster is defined as follows:

Let S be a database schema with rules of the form $S = \{...,R_j = (R_{j1},R_{j2},......,R_{jk}),......\}$

The zero order names derive their values from the domains which are set of atomic values. Let $R_j = (R_{j1},R_{j2},......,R_{jk})$ be a rule in the schema S. Let D_j, D_{j1}, D_{j2} be the domains of R_j, R_{j1}, R_{j2} respectively. Then, the domain D_j is a set of k-tuples generated by the cartesian product of domains $D_{j1}, D_{j2}, ..., D_{jk}$ (i.e. $D_j = (D_{j1} \text{ X } D_{j2} \text{ X ..... X } D_{jk})$). The higher order names such as R_j are set-valued. A cluster is subset of the cartesian product generated by the domains of the names on the right hand side of the rule for the external name in S.

For example, the schema for a hierarchical cluster is given as follows:

```
Domain of
            Sem        is {Fall, Spring, Summer}
            Yr         is int
```

```
            Dept       is Alpha[4]
            CRSE#      is Alphanum[6]
            Id#        is Num[3]
            Instruct   is Alpha[20]
            Studid     is Num[3]
            Studname   is Alpha[20]
            Gr         is { A, B, C, D, F}

RCIS        = (Sem, Yr, Courses)
Courses     = (Dept, Crse#, Instructor, Student)
Instructor  = (Id#, Instruct)
Student     = (Studid, Studname, Gr)
```

In the above example, the domain of Instructor is a set of tuples generated by the domain(Id#) X domain(Instruct). The Figure 1 shows an instance of this cluster.

Registrar								
		Courses						
				Instructor		Student		
Sem	Yr	Dept	Crse#	Id#	Instruct	Studid	Studname	Gr
Fall	89	CS	441	109	Salona, Shyam	042	Hu, Julia	A
						048	Hayte, Anne	B
						052	Smith, P	A
		CS	451	104	Halem, Milt	110	Goff, Roger	A
						112	Virkar, R	B
						052	Smith, P	A
		CS	453	102	Gordon, Chris	099	Oza, Amish	A
						049	Hite, Lee	B
						115	Salgam, R	A
Fall	90	CS	444	111	Wakim, N	042	Hu, Steve	A
				110	Hwang, Steve	155	Waite, Terry	B
						113	Taylor, R	A
		CS	445	109	Salona, Shyam	150	Wang, Paul	A
						032	Berbert, John	B
						052	Smith, P	A
		CS	441	101	Fuller, J	100	Paripati, P	A
						084	Lee, Kyun-He	B
						116	Shah, S	A
						118	Tydings, Will	C

Figure 1 : An Instance of rcis Cluster

An unified query language, called Generalized SQL, is used for defining the cluster schema and also for accessing all the databases. GSQL is an extension of the relational standard SQL. The format of a basic query in GSQL is as follows:

SELECT <Schema_Def>
FROM <Cluster_list>
WHERE <boolean>

The user can query an existing cluster or create a new cluster using the above format. The schema definition represents how the result will be reported or stored, i.e., <schema_def>:..., $(R_{i1}, R_{i2},...,R_{ik})$ as R_i,..., where, the R_{ij}s are the internal names appearing in the cluster_list or higher order names and the R_is are the higher order names. The *from* part of the GSQL query indicates the source cluster definitions, i.e., <cluster_list>: C_1, C_2, ..., where the C_is are the cluster names defined already in the system. In the *where* part the user specifies the boolean conditions, i.e., <boolean>: is a well-formed formula using relational and logical operators and data elements of the system.

Given the cluster data model and GSQL a system for the heterogeneous databases can be implemented [3]. Users can access the data stored in heterogeneous databases using a query language in an uniform way. The query optimizer for such a hereogeneous database system will process the query and generate an optimal execution tree.

The objective of a relational query optimizer is to determine an efficient execution tree for a given SQL query. The operations of the execution tree are generally expressed in the relational algebra. In order to build and optimizer for the GSQL queries involving heterogeneous databases a language akin to the relational algebra but with a scope of heterogeneous databases is required. In the following sections, we introduce such a language to express the primitive operations of heterogeneous databases. The language introduced here is referred to as cluster algebra.

3 The Cluster Algebra

Simply stated, cluster algebra is a generalization of the relational algebra. The cluster algebra operations accept clusters as its operands. A cluster algebra operation performs either selection, projection or join of two clusters at a time. Since, the cluster algebra operations perform one function at a time, the cluster algebra provides a greater flexibility in generating optimal sequence for a query. Unlike a GSQL query, where a user may request a combination of all the above in a single query operation.

3.1 Operations

The cluster algebra operations are classified in two categories: the unary and the binary operations. An unary operation of the cluster algebra requires a single operand, i.e. a cluster. Therefore, it can also be viewed as 1-place function. A binary operation of the cluster algebra requires two clusters as operands. The result of all the cluster algebra operations, which is a cluster, is designated as r. As a part of description of these cluster algebra operations a few new terms are also introduced.

ρ_c: denotes a row of the cluster c

$\rho_c[R_i]$: denotes a value of the attribute R_i of the cluster c

In the subsequent sections, the following cluster algebra operations are introduced and defined.

Unary Operations:

1. Selection
2. Normal Projection
3. Renaming
4. Normalize

Binary Operations:

1. Union
2. Difference
3. Intersection
4. Cross Product
5. Normal Natural Join

Some additional operations, used for restructuring the schema, such as *Embed, Unembed and Expand* are also defined for the cluster algebra. In this work, c[S] is used to denote a cluster c with schema S. Although, it is sufficient to refer a cluster c just by its name, but for the sake of brevity the notation c[S] will be used, where c is name of the cluster and S is the schema of the cluster c. Most of the binary operations require their arguments to be compatible. The concept of compatibility is defined subsequently.

To illustrate the cluster algebra operations in the following sections, rcis_1[RCIS] and rcis_2[RCIS] clusters with the same schema RCIS are defined here. The rules for RCIS schema are as follows:

RCIS = (Sem, Yr, Courses)
Courses = (Dept, Crse#, Instructor, Student)
Instructor = (Id#, Instruct)
Student = (Studid, Studname, Gr)

The Figure 3-1 and Figure 3-2 show instances of the rcis_1 and rcis_2 clusters respectively.

Registrar								
Sem	Yr	Courses						
		Dept	Crse#	Instructor		Student		
				Id#	Instruct	Studid	Studname	Gr
Fall	89	CS	441	101	Fuller, J	041	Hu, Judy	A
				109	Salona, Shyam	049	Hite, Lee	B
						052	Smith, P	A
		CS	442	103	Green, Jim	110	Goff, Roger	A
						112	Virkar, R	B
						052	Smith, P	A
		CS	443	102	Gordon, Chris	099	Oza, Amish	A
						049	Hite, Lee	B
						115	Salgam, R	A
Spring	90	CS	444	111	Wakim, N	043	Henry, J	A
				110	Hwang, Steve	035	Biden, Tom	B
						114	Taylor, Lisa	A
		CS	457	109	Salona, Shyam	150	Wang, Paul	A
						032	Berbert, John	B
						052	Smith, P	A
		CS	458	101	Fuller, J	100	Paripati, P	A
						084	Lee, Kyun-He	B
						116	Shah, S	A
						118	Tydings, Will	C

Figure 3-1: An instance of rcis_1 cluster

Registrar								
Sem	Yr	Courses						
		Dept	Crse#	Instructor		Student		
				Id#	Instruct	Studid	Studname	Gr
Fall	89	CS	441	109	Salona, Shyam	042	Hu, Julia	A
						048	Hayte, Anne	B
						052	Smith, P	A
		CS	451	104	Halem, Milt	110	Goff, Roger	A
						112	Virkar, R	B
						052	Smith, P	A
		CS	453	102	Gordon, Chris	099	Oza, Amish	A
						049	Hite, Lee	B
						115	Salgam, R	A
Fall	90	CS	444	111	Wakim, N	042	Hu, Steve	A
				110	Hwang, Steve	155	Waite, Terry	B
						113	Taylor, R	A
		CS	445	109	Salona, Shyam	150	Wang, Paul	A
						032	Berbert, John	B
						052	Smith, P	A
		CS	441	101	Fuller, J	100	Paripati, P	A
						084	Lee, Kyun-He	B
						116	Shah, S	A
						118	Tydings, Will	C

Figure 3-2: AN instance of rcis_2 cluster

The studinfo[STUDINFO] cluster with relational schema is also used in the examples. The rules of STUDINFO schema are as follows:

STUDINFO = (Studid, Studname, Date_birth, Date_adm)

An instance of studinfo cluster is shown in Figure 3-3.

Studid	Studname	Date_birth	Date_adm
042	Hu, Julia	May,27,1963	Sep,05,1988
048	Hayte,Anne	July,12,1962	Sep,05,1987
052	Smith,P	Jan,04,1963	Sep,05,1988
041	HU,Judy	June,22,1964	Sep,05,1988
049	Hite,Lee	Feb,28,1963	Sep,05,1988
110	Goff,Roger	Sug,30,1965	Sep,05,1989
112	Virkar,R	Oct,2,1959	Sep,05,1988
099	Oza,Amish	Nov,23,1964	Sep,05,1988
115	Salgam,R	Mar,14,1962	Sep,05,1988
100	Paripati,P	Dec, 25,1964	Sep,05,1988
116	Shah,S	Apr,23,1965	Sep,05,1988
150	Wang,Paul	July,14,1966	Sep,05,1989

Figure 3-3: An instance of studinfo cluster

Definition: Let $S_i=(...R_j=(R_{j1},R_{j2},...,R_{jk})...)$ and $S_2=(...\ R_i=(R_{i1},R_{i2},...,R_{in})...)$ be two schemata with higher order names R_i and R_j. The higher order names R_j and R_j are said to be *equal* if following conditions hold.

Let P_i be the set of all zero order names in the R_i and Q_i be the set of all non-zero order names in R_i. Let P_j be the set of all zero order names in the R_j and Q_j be the set of all non-zero order names in R_j.

I. $Q_i = \varnothing \Leftrightarrow Q_j = \varnothing$ and $A_k \in P_i \Leftrightarrow A_k \in P_j$ where A_k has $dom(A_k)$

II. If $Q_i \neq \varnothing$ then

A. $Q_i \neq \varnothing \Leftrightarrow Q_j \neq \varnothing$ and

$A_k \in Q_i \Leftrightarrow A_k \in Q_j$ such that

$A_k \in Q_i$ and $A_k \in Q_j$ are equal.

B. $A_k \in P_i \Leftrightarrow A_k \in P_j$ where

A_k *has a* $dom(A_k)$.

For example, let RCIS_1 = (Sem, Yr, Courses) and RCIS_2 = (Sem, Courses, Yr) be two database schema where Courses= (Dept, Crse#, Instruct). The higher order name RCIS_1 and RCIS_2 are equal.

Definition: Let $c[S_1]$ and $d[S_2]$ be two clusters with schemata S_1 and S_2 respectively. The clusters $c[S_1]$ and $d[S_2]$ are considered *Compatible* if and only if either S_1 and S_2 are equal or there exists one-to-one and onto mapping f such that:

1. For each $A_i \in S_1$ there exists an $A_j \in S_2$ such that $A_j = f(A_i)$ and $A_i = f^{-1}(A_j)$ where A_i and A_j are the zero-order names and have the same domain.

2. For each $B_i \in S_1$ there exists a $B_j \in S_2$ such that $B_j = f(B_i)$ and $B_i = f^{-1}(B_j)$. Also, the higher order names B_i and B_j are *compatible.*

For example, let RCIS_1 and RCIS_2 be two database schema defined as follows:

RCIS_1 = (Sem, Yr, Courses, Student)
Courses= (Dept, Crse#, Credits)
Student= (Studid, Name, Grade);

RCIS_2 = (Sem, Yr, Student, Course1)
Course1= (Dept, Credit, Course#);

The example schemata defined above are compatible but not equal.

3.1.1 Selection

The selection operation is used to choose some rows from the cluster based on a criteria or condition. The condition is also referred to as a formula. The formal definition of a formula is given as follows:

Formula Definition:

Let A_i be an attribute i in a database schema S. A *formula* F in the schema S is defined as:

1. Let $K \in Dom(A_i)$ be a constant, then A_i *<op> Const* is a formula, where, $<op> \in \{ = \; \neq \; > \; \geq \; < \; \leq \}$.

2. Let *X* and *Y* be formulas, then the $X \wedge Y$, $X \vee Y$, and $\neg X$ are formulas.

3. Nothing else is a formula.

Definition: Let *c[S]* be a cluster with a schema *S* and ρ_c be a cluster row in *c*. Then, the *selection* of a cluster *c[S]*, denoted by $\sigma_F(c[S])$, under a formula *F* is a subset of *c[S]* such that:

$$\sigma_F(c[S]) = \{ \rho_r \mid \rho_r \in c \text{ and } F \text{ is true for } \rho_r \}$$

For examples, the result of the operation, $rcis\_sel = \sigma_{crse\# = '441'}(rcis\_1)$, is shown in Figure 3-4.

Registrar								
		Courses						
				Instructor		Student		
Sem	Yr	Dept	Crse#	Id#	Instruct	Studid	Studname	Gr
Fall	89	CS	441	101	Fuller, J	041	Hu, Judy	A
						049	Hite, Lee	B
				109	Salona, Shyam	052	Smith, P	A

Figure 3–4: rcis_sel cluster

3.1.2 Normal Projection

Normal projection operation is used for chosing columns from a cluster. The result of a normal projection operation is always a relational cluster. A non-relational cluster can be constructed from the relational cluster by making use of the *embed* and *unembed* operations.

Definition: Let *c[S]* be a cluster with schema *S*. The *normal projection* of *c[S]* over *P* where *P* is defined subsequently, denoted as $\pi^n_P(c[S])$, is computed as follows:

$$P = \{ R_i \mid R_i \text{ is a zero order name and } R_i \in S \}$$

$$\pi^n_P[c] = \{\rho_r \mid \forall R_i \in P \quad \rho_r[R_i] = \rho_c[R_i] \text{ and } \rho_c \in c \}$$

For example, the normal projection of the rcis_sel cluster on attributes Sem, Yr, Dept, Crse#, Instruct, Studname, Gr is shown in Figure 3-5.

Sem	Yr	Dept	Crse#	Instruct	Studname	Gr
Fall	89	CS	441	Fuller, J	Hu, Judy	A
Fall	89	CS	441	Fuller, J	Hite, Lee	B
Fall	89	CS	441	Fuller, J	Smith, P	A
Fall	89	CS	441	Salona, Shyam	Hu, Judy	A
Fall	89	CS	441	Salona, Shyam	Hite, Lee	B
Fall	89	CS	441	Salona, Shyam	Smith, P	A

Figure 3–5: rcis_proj cluster

3.1.3 Renaming

Let c[S] be a cluster with the schema S and let A be an attribute of S. Also, let B be an attribute such that $B \not\in (S - A)$ the renaming operation is defined as follows:

1. If A is a zero-order name with the Domain(A) = Domain(B). The *renaming* of A to B in the cluster c[S], denoted by $\delta_{A \leftarrow B}(c[S])$, is a cluster with schema S' = (S - A) U B such that the column A of S is replaced by the column B. The result cluster of the operation denoted by r is derived as follows:

$$\delta_{A \leftarrow B}\ (c[S]) = \{\ \rho_r \ \mid\ \rho_c \in c[S],\ \rho_r[S-A] = \rho_c[S-A] \ \wedge\ \rho_r[B] = \rho_c[A]\}$$

2. If A = $(A_1,A_2,...,A_n)$ and B= $(B_1,B_2,...,B_n)$ are higher order names such that A and B are *compatible* then, let *f* be the compatibility function such that $B_j = f(A_i)$. The *renaming* of A to B in the cluster c[S], denoted by $\delta_{A \leftarrow B}(c[S])$, is a cluster r with schema S' = (S - A) U B such that column A in S is replaced by column B. The cluster r is derived as follows:

   2.1 For all A_i in A

$$\delta_{A \leftarrow f(A_i)}(c[S] = \{\ \rho_r \ \mid\ \rho_c \in\ c[S] \ \wedge$$
$$\rho_r[S - A_i] = \rho_c[S - A_i] \ \wedge\ \rho_r[f(A_i)] = \rho_c[A_i]\}$$

2.2 Replace name A by name B in S'.

3.1.4 Expand

Expand is a macro that takes any zero-order or non-zero order name as input and generates a equivalent expanded version for that name. The expand is not an algebraic operation. Let c[S] be a cluster with schema S and let A be a name in S. The expand operation is defined as follows:

1. If A is a zero-order name then, expand(A) = A .

2. If $A = (A_1, A_2, ..., A_m)$ is a higher order name in the schema S then,
   expand(A) = $A_1, A_2, ..., A_m$

For example, the result of expand(STUDINFO) is as follows:

expand(STUDINFO) = Studid, Studname, Date_birth, Date_adm

3.1.5 Embed

The *embed* is an operator that alters the structural relationships amongst rules of the schema. This operator is similar to the nest operator defined by Roth [10] except that the nest operator is defined for the relational and hierarchical databases only.

Definition: Let *c[S]* be a cluster with schema *S*. The embed operator, as the name implies, embeds a subset of attributes into the another subset of the attributes of the schema *S*. All cluster rows are suitably modified to reflect the change in the schema.

Let R_i be a non zero-order name in S and $A = \{A_1, A_2, ..., A_m\}$ be an attribute set which is a subset of attributes of R_i. Also, the schema S does not have any rule with $(A_1, A_2, ..., A_m)$ on its right hand side. Define a set of attributes $B = \{B_1, B_2, ..., B_m\} = R_i - A$.

The embed operation is denoted by $\eta(A,B,S_i') = c'[S']$ where S' is generated as follows:

I. Define a rule $R_i' = (B_1, B_2, ..., B_k, A = (A_1, A_2, ..., A_m))$

II. Also, define a new rule $A = (A_1, A_2, ..., A_m)$

III. Replace the rule for R_i in S by the new rule $R_i' = (\{B_1, B_2, ..., B_k, A)$ in S. Substitute the name S by S'.

c' is generated as follows:

$$c' = \left\{ \rho_r \mid \exists \rho_c \in c : \bigvee_{B_i \in B} \rho_r[B_i] = \rho_c[B_i] \wedge \rho_r[A] = \left\{ \rho_{c'}[A] \mid \exists \rho_{c'} \in c \ni \bigvee_{B_i \in B} \rho_{c'}[B_i] = \rho_c[B_i] \right\} \right\}$$

For example, the result of embed operation, $\eta(A, B, rcis\_proj)$, where A = { Studname, Gr } and B = {Sem, Yr, Dept, crse#, Instruct } is shown in Figure 3-6.

Sem	Yr	Dept	Crse#	Instruct	A	
					Studname	Gr
Fall	89	CS	441	Fuller, J	Hu, Judy	A
					Hite, Lee	B
					Smith, P	A
Fall	89	CS	441	Salona, Shyam	Hu, Judy	A
					Hite, Lee	B
					Smith, P	A

Figure 3-6: rcis_embed cluster

3.1.6 Unembed

The unembed operation accomplishes the inverse of the embed operator. Simply stated, this operation flattens an embedded table. In other words, for a database cluster that has just one table embedded with in the external table, the single unembed operation will produce a relational cluster.

Definition: Let c[S] be a cluster with schema S. Let $R_i = (R_{i1}, R_{i2}, \ldots, R_{im})$ be a rule in S. Also, let $R_{ij} \in R_i$ be a non-zero order name such that $R_{ij} = (R_{ij1}, R_{ij2}, \ldots, R_{ijn})$ is a rule in S. The unembed operation $\phi(R_i, R_{ij}, c[S])$ = c'[S'] is defined as follows:

S' is derived from S by

Case I. R_{ij} appears only on the right hand side of a rule R_i in S.

i) Copy all the rules from S to S'.

ii) Modify the right hand side of the R_i in S' to R_i = $(R_{i1}, R_{i2}, \ldots, R_{ij1}, R_{ij2}, R_{ij3}, \ldots, R_{ijn}, \ldots, R_{im})$.

iii) Drop the rule $R_{ij} = (R_{ij1}, R_{ij2}, \ldots, R_{ijn})$ from S'.

Case II. R_{ij} appears on the right hand side of a rule R_j in S such that $R_i \neq R_j$ and $R_j = (R_{j1}, R_{j1}, \ldots, R_{ji}, \ldots, R_{jm})$.

i) Copy all the rules from S to S'

ii) Modify the right hand side of the R_i in S' to R_i = $(R_{i1}, R_{i2}, \ldots, R_{ij1}, R_{ij2}, R_{ij3}, \ldots, R_{ijn}, \ldots, R_{im})$.

iii) Replace R_{ij} by R_i in all the R_j rules where R_{ij} appears on the right hand side. Thus R_j rule becomes $Rj = (R_{j1}, R_{j2}, ..., R_i, ..., R_{im})$.

iv) Drop the rule where R_{ij} appears on the left hand side.

c' is derived from c by

Let $t(R_{ij})$ denote a table row of R_{ij} in R_i. Let $t_{pr}(R_{ij}, R_i)$ denote the parent row of $t(R_{ij})$ in R_i.

For each table row $t(R_{ij})$ generate a new table row, t_r, of R_i as follows:

I. Let $t^* = t_{pr}(R_{ij}, R_i)$ be the parent row.

II. $\forall \quad R_{ik} \in R_i \quad \wedge \quad R_{ik} \notin R_{ij} \quad then \quad t_r[R_{ik}] = t^*[R_{ik},]$

III. $\forall \quad R_{ik} \in R_{ij} \quad t_r[R_{ik}] = t_{ij}[R_{ik}]$

IV. For all R_j such that R_{ij} appears on the right hand side and $R_j \neq R_i$

$$t_{pr}(R_i, R_j) = t_{pr}(R_{ij}, R_j)$$

For example, the result of unembed operation, $\phi(B, A, rcis\_emb)$, is given in Figure 3-7.

Sem	Yr	Dept	Crse#	Instruct	Studname	Gr
Fall	89	CS	441	Fuller, J	Hu, Judy	A
Fall	89	CS	441	Fuller, J	Hite, Lee	B
Fall	89	CS	441	Fuller, J	Smith, P	A
Fall	89	CS	441	Salona, Shyam	Hu, Judy	A
Fall	89	CS	441	Salona, Shyam	Hite, Lee	B
Fall	89	CS	441	Salona, Shyam	Smith, P	A

Figure 3–7: rcis_unemb cluster

3.1.7 Normalize

The normalize operation transforms any heterogeneous database cluster to a database cluster with relational schema.

Definition: Let c[S] be a cluster with schema S. The operation *normalize*, N(c[S]), generates an equivalent cluster with a relational schema.

The algorithm to derive the normalized relational cluster r[S'] from the original cluster c[S] is as follows:

Step 1. Set r[S'] = c[S].

Step 2. Repeat {

Decompose S' in subsets A and B such that

$S' = A \cup B$ and

A is a set of all zero-order names in S' and

B is a set of all non-zero order names in S'.

If B is not empty then,

$\forall_{B_i \in B} \; r[S'] = N(unembed(S', B_i, c'[S']))$

} until B is an empty set

3.1.8 Union

The union operation is binary operation on the cluster operands. The operands of an union operation are always compatible.

Definition: Let $c[S_1]$ and $d[S_2]$ be two compatible clusters. The *union* of the two compatible clusters c U d is a cluster consisting of the rows belonging to either c or d or both. The result of the union operation r[S] is derived as follows:

1. S_1 and S_2 are compatible so either $S_1 = S_2$ or generate S_2' by renaming all the attributes in $(S_2 - S_1) \cap S_2$ to $(S1 - S_2) \cap S_1$ such that $S_1 = S_2'$.

2. Let P be the subset of the S_1 such that $P = \{A_i \mid A_i \in S_1 \wedge A_i \text{ is a zero order name}\}$ and let Q be the subset of the S_1 such that $Q = \{B_i \mid B_i \in S_1 \wedge B_i \notin P\}$, define,

case I. $\exists \rho_c \in c$

$$\forall \rho_d \in d \ \exists A_i \ni \rho_c[A_i] \neq \rho_d[A_i] \text{ then,}$$

$$c \cup d = \{\rho_r \mid \rho_r = \rho_c\}$$

case II. $\exists \rho_d \in d$

$$\forall \rho_c \in c \ \exists A_i \ni \rho_d[A_i] \neq \rho_c[A_i] \text{ then,}$$

$$c \cup d = \{\rho_r \mid \rho_r = \rho_d\}$$

case III. $\exists \rho_c \in c \text{ and } \exists \rho_d \in d$

$$\forall A_i \ \rho_d[A_i] = \rho_c[A_i] \text{ then,}$$

$$c \cup d = \{\rho_r \mid \forall A_i \ \rho_r[A_i] = \rho_c[A_i] \wedge \forall B_i \ \rho_r[B_i] = \rho_c[B_i] \cup \rho_d[B_i]\}$$

Whenever Q is an empty set, it is an usual union of relations.

For example, the result of union of rcis_1 and rcis_2 clusters denoted by rcis_union = rcis_1 U rcis_2 is shown in Figure 3-8.

Registrar								
		Courses						
				Instructor		Student		
Sem	Yr	Dept	Crse#	Id#	Instruct	Studid	Studname	Gr
Fall	89	CS	441	101	Fuller, J	042	Hu, Julia	A
				109	Salona, Shyam	048	Hayte, Anne	B
						052	Smith, P	A
						041	Hu, Judy	A
						049	Hite, Lee	B
		CS	451	104	Halem, Milt	110	Goff, Roger	A
						112	Virkar, R	B
						052	Smith, P	A
		CS	453	102	Gordon, Chris	099	Oza, Amish	A
						049	Hite, Lee	B
						115	Salgam, R	A
		CS	451	104	Halem, Milt	110	Goff, Roger	A
						112	Virkar, R	B
						052	Smith, P	A
		CS	453	102	Gordon, Chris	099	Oza, Amish	A
						049	Hite, Lee	B
						115	Salgam, R	A
Fall	90	CS	444	111	Wakim, N	042	Hu, Steve	A
				110	Hwang, Steve	155	Waite, Terry	B
						113	Taylor, R	A
		CS	445	109	Salona, Shyam	150	Wang, Paul	A
						032	Berbert, John	B
						052	Smith, P	A
		CS	441	101	Fuller, J	100	Paripati, P	A
						084	Lee, Kyun-He	B
						116	Shah, S	A
						118	Tydings, Will	C
Sprng	90	CS	444	111	Wakim, N	043	Henry, J	A
				110	Hwang, Steve	035	Biden, Tom	B
						114	Taylor, Lisa	A
		CS	457	109	Salona, Shyam	150	Wang, Paul	A
						032	Berbert, John	B
						052	Smith, P	A
		CS	458	101	Fuller, J	100	Paripati, P	A
						084	Lee, Kyun-He	B
						116	Shah, S	A
						118	Tydings, Will	C

Figure 3-8: rcis_union cluster

3.1.9 Intersection

The intersection operation of clusters accepts two compatible clusters as input and produces a cluster.

Definition: Let $c[S_1]$ and $d[S_2]$ be two compatible clusters. The *intersection* of the two compatible clusters, $c \cap d$, is a cluster consisting of the rows which belong to both clusters c and d. The result of the intersection operation is derived as follows:

1. S_1 and S_2 are compatible so either $S_1 = S_2$ or generate S_2' by renaming all the attributes in $(S_2 - S_1) \cap S_2$ to $(S1 - S_2) \cap S_1$ such that $S_1 = S_2$'.

2. Let P be the subset of the S_1 such that $P = \{A_i \mid A_i \in S_1 \wedge A_i \text{ is a zero order name}\}$ and let Q be the subset of the S_1 such that $Q = \{B_i \mid B_i \in S_1 \wedge B_i \notin P\}$, define,

The result of intersection is generated as follows:

$$\exists \rho_c \in c, \exists \rho_d \in d \ni \forall A_i \ \rho_c[A_i] = \rho_d[A_i] \text{ and}$$
$$\forall B_i \ \rho_c[B_i] \cap \rho_d[B_i] \neq \phi \text{ then,}$$
$$c \cap d = \{\rho_r \mid \forall A_i \ \rho_r[A_i] = \rho_c[A_i] \wedge$$
$$\forall B_i \ \rho_r[B_i] = \rho_c[B_i] \cap \rho_d[B_i]\}$$

The relational intersection is a special case of above definition where Q is an empty set.

For example, the result of intersection of the rcis_1 and rcis_2 cluster denoted by *rcis_intersect* = $rcis\_1 \cap rcis\_2$ is shown in Figure 3-9.

Registrar								
		Courses						
				Instructor		Student		
Sem	Yr	Dept	Crse#	Id#	Instruct	Studid	Studname	Gr
Fall	89	CS	441	101	Fuller, J	052	Smith, P	A

Figure 3-9: rcis_intersect cluster

3.1.10 Difference

The difference operation of clusters is a binary operation, that accepts two compatible clusters as arguments. The result of the operation is a cluster.

Definition: Let $c[S_1]$ and $d[S_2]$ be two compatible clusters. The *difference* of the two clusters, written as c - d, retains those rows of the cluster c that are not in the cluster d. It is defined as follows:

1. S_1 and S_2 are compatible so either $S_1 = S_2$ or generate S_2' by renaming all the attributes in $(S_2 - S_1) \cap S_2$ to $(S1 - S_2) \cap S_1$ such that $S_1 = S_2$'.

2. Let P be the subset of the S_1 such that $P = \{A_i \mid A_i \in S_1 \wedge A_i \text{ is a zero order name}\}$ and let Q be the subset of the S_1 such that $Q = \{B_i \mid B_i \in S_1 \wedge B_i \notin P\}$, define,

Result of difference operation is generated as follows:

Case I. $\exists \; \rho_c \in c, \; \exists \; \rho_d \in d \ni$

$$\forall \; A_i \;\; \rho_c[A_i] = \rho_d[A_i]$$

then,

$$c - d = \{ \rho_r \mid \forall \; A_i \;\; \rho_r[A_i] = \rho_c[A_i] \text{ and }$$
$$\forall \; B_i \;\; \rho_r[B_i] = \rho_c[B_i] - \rho_d[B_i] \}$$

Case II. $\exists \; \rho_c \in c \ni \forall \; \rho_d \; \exists \; A_i \; \rho_c[A_i] \neq \rho_d[A_i] \;\}$

then,

$$c - d = \{ \rho_r \mid \rho_r = \rho_c \}$$

Whenever Q is an empty set, it is the usual difference of relations.

For example, the difference of rcis_1 and rcis_2 clusters, rcis_diff = rcis_1 - rcis_2, is a cluster shown in Figure 3-10.

Registrar								
		Courses						
				Instructor		Student		
Sem	Yr	Dept	Crse#	Id#	Instruct	Studid	Studname	Gr
Fall	89	CS	441	101	Fuller, J	041	Hu, Judy	A
						049	Hite, Lee	B
		CS	442	103	Green, Jim	110	Goff, Roger	A
						112	Virkar, R	B
						052	Smith, P	A
		CS	443	102	Gordon, Chris	099	Oza, Amish	A
						049	Hite, Lee	B
						115	Salgam, R	A
Sprng	90	CS	444	111	Wakim, N	043	Henry, J	A
						035	Biden, Tom	B
				110	Hwang, Steve	114	Taylor, Lisa	A
		CS	457	109	Salona, Shyam	150	Wang, Paul	A
						032	Berbert, John	B
						052	Smith, P	A
		CS	458	101	Fuller, J	100	Paripati, P	A
						084	Lee, Kyun-He	B
						116	Shah, S	A
						118	Tydings, Will	C

Figure 3-10: rcis_diff cluster

3.1.11 Cross Product

The cross product of clusters is a binary operation. The cross product requires that the schemata of the two clusters do not have qualified duplicate names. The result of the operation is a cluster.

Definition: Let $c[S_1]$ and $d[S_2]$ be two clusters. Also, $S_1 \cap S_2 = \phi$. The *cross product* of the two clusters, c X d , is defined by

$c[S_1] \text{ X } d[S_2] = r[S']$ where

S' is derived from S_1 and S_2 by the following algorithm:

1. Copy all rules from S_1 to S'.

2. Copy all rules from S_2 to S'.

3. Add a new rule S' = (expand(S_1), expand(S_2))

4. Drop the rules corresponding to S_1 and S_2 on the left hand side.

r is derived from c and d by following algorithm

For each $\rho_c \in c$

  For all $\rho_d \in d$

    For all $A_i \in S'$

$$If \quad A_i \in S_1 \quad \rho_r[A_i] = \rho_c[A_i]$$
$$If \quad A_i \in S_2 \quad \rho_r[A_i] = \rho_d[A_i]$$

3.1.12 Normal Natural Join

The operation of normal natural join is an equijoin of two clusters on all the common attributes in their normalized schema. The normal natural join retains only one of such duplicate columns.

Definition: Let $c[S_1]$ and $d[S_2]$ be two clusters with schemata S_1 and S_2 respectively. The *normal natural join* , $r[S'] = c$ *NJN* d, is defined as follows:

Let

$$X = \{ R_i \mid R_i \in S_1 \wedge R_i \text{ is a zero order name} \}$$

Let

$$Y = \{ R_i \mid R_i \in S_2 \wedge R_i \text{ is a zero order name} \}$$

Let $Z = X \cap Y$ then,

$S' = (X,Z,Y)$

r is derived from c' and d' as follows:

$$r' = \left\{ \rho_r \mid \exists \rho_c \in c, \rho_d \in d \ni \bigvee_{Z_i \in Z} \rho_c[Z_i] = \rho_d[Z_i] : \rho_r[X] = \rho_c[X] \wedge \rho_r[Y-Z] = \rho_d[Y-Z] \right\}$$

For example, the result of rcis_1 NJN studinfo is shown in Figure 3-11.

Sem	Yr	Dept	Crse#	Id#	Instruct	Studid	Studname	Gr	Date_birth	Date_adm
Fall	89	CS	441	101	Fuller,J	041	Hu,Judy	A	June,22,1964	Sep,05,1988
Fall	89	CS	441	101	Fuller,J	049	Hite,Lee	B	Feb,28,1963	Sep,05,1988
Fall	89	CS	441	101	Fuller,J	052	Smith,P	A	Jan,04,1963	Sep,05,1988
Fall	89	CS	441	109	Salona,S	041	Hu,Judy	A	June,22,1964	Sep,05,1988
Fall	89	CS	441	109	Salona,S	049	Hite,Lee	B	Feb,28,1963	Sep,05,1988
Fall	89	CS	441	109	Salona,S	052	Smith,P	A	Jan,04,1963	Sep,05,1988
Fall	89	CS	442	103	Green,Jim	110	Goff,Roger	A	Aug,30,1965	Sep,05,1989
Fall	89	CS	442	103	Green,Jim	112	Virkar,R	B	Oct,02,1959	Sep,05,1988
Fall	89	CS	442	103	Green,Jim	052	Smith,P	A	Jan,04,1963	Sep,05,1988
Fall	89	CS	443	102	Gordon,C	099	Oza,Amish	A	Nuv,23,1964	Sep,05,1988
Fall	89	CS	443	102	Gordon,C	049	Hite,Lee	B	Feb,28,1963	Sep,05,1988
Fall	89	CS	443	102	Gordon,C	115	Salgam,R	A	May,14,1952	Sep,05,1988
Spring	89	CS	457	109	Salona,S	150	Wang,Paul	A	July,14,1966	Sep,05,1989
Spring	89	CS	457	109	Salona,S	052	Smith,Paul	A	Jan,04,1963	Sep,05,1988
Spring	89	CS	458	101	Fuller,J	100	Paripati,P	A	Dec,25,1964	Sep,05,1988
Spring	89	CS	458	101	Fuller,J	116	Shah,S	A	Apr,23,1965	Sep,05,1988

Figure 3–11: rcis_studinfo_join cluster

4 Conclusion

The heterogeneous data model used in this work is a suitable choice for storing complex objects compared to the relational model. This data model allows integration of the data stored in hierarchical and network data models. The query languages developed for this data model, GSQL and GCALC, permit uniform access to the data stored in the relational as well as the non relational data models. However, these query languages are not suitable for the query optimization purposes. The cluster algebra has been defined to fill this void. The cluster algebra operators perform one single step of the query at a time. This provides greater flexibility of rearranging and restructuring the sequence of operators needed to answer a query. Various extensions of relational algebra proposed [2,10] for the recursive extensions of relational model usually encompass the data stored in PNF [10] or similar formats. These extensions have the scope of the relational and the hierarchical data models. The cluster algebra proposed here embodies the uniform access mechanism for the data stored in the relational, hierarchical and the network data models.

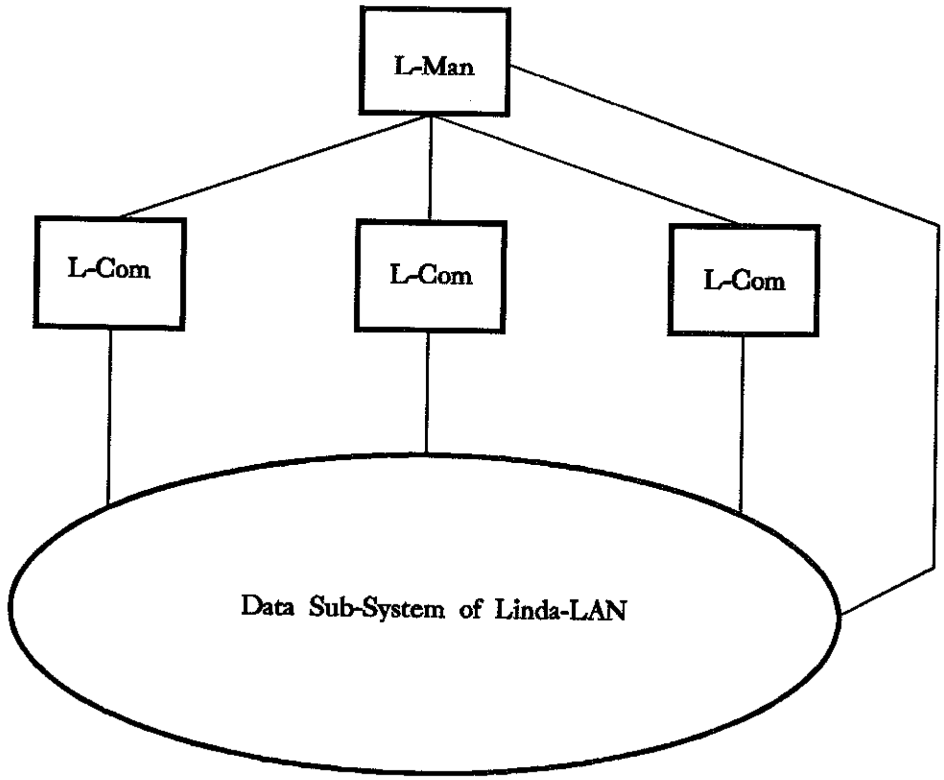

Figure 3. Linda-LAN as viewed by the Control Sub-System

References

1 Abiteboul, S., and Bidoit, N., "Non First Normal Form Relations to Represent Hierarchically Organized Data" in Proceedings of the third ACM *SIGACT-SIGMOD symposium on Database systems,* Waterloo, April 1984, pp. 191-200

2 Arisawa, H.,Moriya, K., and Miura, T., "Operations and the Properties on Non First Normal Form Relational Databases" in *Proceedings of Ninth International Conference on Very Large Databases,* Florence, Oct. 1983, pp. 197-204

3 Bhasker, B. Huang, S and Jacobs, B., "DAVID: NASA's Heterogeneous Distributed Database Management System" in *Proceedings of the 1990 Symposium on Applied Computing,* April 1990

4 Graham, M. H., "On universal relation" *Technical Report, Dept. of Computer Science, University of Toronto,* Toronto, Ontario, Canada, Sept. 1979

5 Gupta, A. (Ed.), "Integration of Information Systems: Bridging Heterogeneous Databases" *IEEE Press, 1989*

6 Hull, R. and Yap, C., "The Format Model: A Theory of Database Organization" *Journal of the ACM* Vol. 31, No. 3, July 1984, pp. 518-537

7 Jacobs, B. E., "On Database Logic" *Journal of the ACM, Vol. 29, No.2, Apr. 1982, pp. 310-332*

8 Kuper, G. and Vardi, M., "A New Approach to Database Logic" in *Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems,* Apr. 1984, ACM , New York, 1984, pp. 86-96

9 Makinouchi, A., "A Consideration on Normal Form of Not Necessarily Normalized Relation in the Relational Data Model" in *Proceedings of the Third International conference on Very Large Databases,* Oct. 1977, pp. 447-453

10 Roth, M., Korth, H. and Silberschatz, A., "Extended Algebra and Calculus for Nested Relational Databases" *ACM Transactions on Database Systems,* Vol. 13, No. 4, Dec. 1988, pp. 389-417

11 Smith , J.M., and Smith, D.C.P., "Database Abstractions: Aggregation and Generalization" *ACM Transaction on Database Systems*, Vol. 12, No. 2, June 77, pp. 105-133

12 Sheth A., and Larson, J., "Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases" *ACM Computing Surveys,* Vol. 22, No. 3, Sept 1990.