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.
1 Introduction ................................................................................................. 1

2 The Cluster Data Model ............................................................................. 1

3 The Cluster Algebra ................................................................................... 7
  3.1 Operations .............................................................................................. 7
    3.1.1 Selection ......................................................................................... 12
    3.1.2 Normal Projection ......................................................................... 13
    3.1.3 Renaming ......................................................................................... 14
    3.1.4 Expand ............................................................................................ 15
    3.1.5 Embed ............................................................................................. 15
    3.1.6 Unembed ......................................................................................... 17
    3.1.7 Normalize ......................................................................................... 19
    3.1.8 Union .............................................................................................. 20
    3.1.9 Intersection .................................................................................... 22
    3.1.10 Difference .................................................................................... 23
    3.1.11 Cross Product ............................................................................. 24
    3.1.12 Normal Natural Join .................................................................... 25

4 Conclusion ................................................................................................. 27

References .................................................................................................... 28
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 $HS = \{..., R_j = (R_{j1}, R_{j2}, \ldots, R_{jk}), \ldots\}$ 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\(^1\) \((R_0, R_1, \ldots, R_i, R_{i+1}, \ldots, R_r)\) 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 \(N = \{..., R_j = (R_{j1}, R_{j2}, \ldots, R_{jk}), \ldots\}\)

where:

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

ii. For all valid subcluster access paths\(^2\) \((R_0, R_1, \ldots, R_i, R_{i+1}, \ldots, R_r)\) 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 \(S = \{..., R_j = (R_{j1}, R_{j2}, \ldots, R_{jk}), \ldots\}\) be a collection of rules. A Subcluster Access Path in \(S\) is a list of the form:

\((R_0, R_1, \ldots, R_i, R_{i+1}, \ldots, R_r)\)

where, for \(i = 0, 1, \ldots, r\), the following hold:

i. \(R_0\) 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. \( D_j = (D_{j1} \times D_{j2} \times ... \times D_{jk}) \)). 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:

\[
\text{Domain of} \quad \begin{array}{c}
\text{Sem} \\
\text{Yr}
\end{array} \quad \text{is} \quad \{ \text{Fall, Spring, Summer} \} \\
\text{is int}
\]

4
Dept is Alpha[4]
CRSE# is Alphanumeric[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.

<table>
<thead>
<tr>
<th>SemYr</th>
<th>Dep</th>
<th>Crse#</th>
<th>Id#</th>
<th>Instructor</th>
<th>Studid</th>
<th>Studname</th>
<th>Gr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall 89</td>
<td>CS</td>
<td>441</td>
<td>109</td>
<td>Salona, Shyam</td>
<td>042</td>
<td>Hu, Julia</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>451</td>
<td>104</td>
<td>Halem, Milt</td>
<td>048</td>
<td>Hayte, Anne</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>453</td>
<td>102</td>
<td>Gordon, Chris</td>
<td>110</td>
<td>Coff, Roger</td>
<td>A</td>
</tr>
<tr>
<td>Fall 90</td>
<td>CS</td>
<td>444</td>
<td>111</td>
<td>Wakim, N</td>
<td>112</td>
<td>Virkar, R</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>445</td>
<td>109</td>
<td>Salona, Shyam</td>
<td>049</td>
<td>Hite, Lee</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>441</td>
<td>101</td>
<td>Fuller, J</td>
<td>084</td>
<td>Lee, Kyun-He</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>116</td>
<td>Shah, S</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>118</td>
<td>Tydings, Will</td>
<td>C</td>
</tr>
</tbody>
</table>

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_ijk'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 heterogeneous 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 a 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.

\[ \rho_c \]: 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.
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.

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

Figure 3–3: An instance of studinfo cluster

Definition: Let \( S_1 = (...)R_j= (R_{j1}, R_{j2}, ..., R_{jk}) ... \) and \( S_2 = (...)R_i= (R_{i1}, R_{i2}, ..., R_{in}) ... \) be two schemata with higher order names \( R_j \) and \( R_i \). The higher order names \( R_j \) and \( R_i \) 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. \( Q_i = \emptyset \iff Q_j = \emptyset \) and \( A_k \in P_i \iff A_k \in P_j \), where \( A_k \) has \( \text{dom}(A_k) \)

II. If \( Q_i \neq \emptyset \) then
A. \( Q_i \neq \emptyset \iff Q_j \neq \emptyset \) and
\[ A_k \in Q_i \iff A_k \in Q_j \] such that
\[ A_k \in Q_i \text{ and } A_k \in Q_j \] are equal.

B. \( A_k \in P_i \iff A_k \in P_j \) where
\[ A_k \text{ has a } \text{dom}(A_k). \]

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 $K \in \text{Dom}(A_i)$ be a constant, then $A_i <op> \text{Const}$ 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 $\rho_c$ 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, $rcis\_sel = \sigma_{grse\#1,441}(rcis\_1)$, is shown in Figure 3-4.
3.1.2 Normal Projection

Normal projection operation is used for choosing 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 πₚ(S), is computed as follows:

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

\[ \pi_p[c] = \{ \rho_r \mid \forall R_i \in P \quad \rho_r[R_i] = \rho_c[R_i] \quad \text{and} \quad \rho_c \in c \} \]

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.
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 \( B \notin (S - A) \) 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 \( \delta_{A \rightarrow B}(c[S]) \), is a cluster with schema \( S' = (S - A) \cup B \) 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:

\[
\delta_{A \rightarrow B}(c[S]) = \{ \rho_r \mid \rho_e \in c[S], \rho_r[S - A] = \rho_e[S - A] \land \rho_r[B] = \rho_e[A] \}
\]

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 \( \delta_{A \rightarrow B}(c[S]) \), is a cluster with schema \( S' = (S - A) \cup B \) such that column \( A \) in \( S \) is replaced by column \( B \). The cluster \( r \) is derived as follows:

\[\delta_{A \rightarrow f(A_i)}(c[S]) = \{ \rho_r \mid \rho_e \in c[S] \land \rho_r[S - A_i] = \rho_e[S - A_i] \land \rho_r[f(A_i)] = \rho_e[A_i] \}\]
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 \( \eta(A, B, S') = c'[S'] \) where \( S' \) is generated as follows:

I. Define a rule \( R_i = (B_1 B_2 \ldots B_m, A = (A_1 A_2 \ldots A_m)) \)

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

III. Replace the rule for \( R_i \) in \( S \) by the new rule \( R_i = (B_1 B_2 \ldots B_m A) \) in \( S \). Substitute the name \( S \) by \( S' \).

\( c' \) is generated as follows:

\[
c' = \left\{ \rho_r \mid \exists \rho_c \in c : \bigvee_{\beta \in \theta} \rho_r[B_\beta] = \rho_c[B_\beta] \land \rho_r[A] = \left\{ \rho_c[A] \mid \exists \rho_c' : c \ni \bigvee_{\beta \in \theta} \rho_c'[B_\beta] = \rho_c[B_\beta] \right\} \right\}
\]

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

<table>
<thead>
<tr>
<th>Sem</th>
<th>Yr</th>
<th>Dept</th>
<th>Crse#</th>
<th>Instruct</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>89</td>
<td>CS</td>
<td>441</td>
<td>Fuller, J</td>
<td>Hu, Judy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Hite, Lee</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Smith, P</td>
<td>A</td>
</tr>
<tr>
<td>Fall</td>
<td></td>
<td></td>
<td>441</td>
<td>Salona, Shyam</td>
<td></td>
</tr>
<tr>
<td></td>
<td>89</td>
<td>CS</td>
<td>441</td>
<td></td>
<td>Hu, Judy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Hite, Lee</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Smith, P</td>
<td>A</td>
</tr>
</tbody>
</table>

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 $R_i = (R_{i1}, R_{i2}, \ldots, R_{ij}, R_{ij2}, \ldots, R_{ijn}, \ldots, R_{im})$.

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 $R_j = (R_{j1}, R_{j2}, \ldots, R_{jm})$.

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

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

17
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 $R_j = (R_{j1}, R_{j2}, ..., R_{j\beta}, ..., R_{jm})$.

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

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

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, $\phi(B, A, rcis_\text{emb})$, is given in Figure 3-7.
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
    \[ S' = A \cup B \]
    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[S₁] and d[S₂] 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. S₁ and S₂ are compatible so either S₁ = S₂ or generate S₂' by renaming all the attributes in (S₂ - S₁) ∩ S₂ to (S₁ - S₂) ∩ S₁ such that S₁ = S₂'.

2. Let P be the subset of the S₁ such that
   \[ P = \{ A_i \mid A_i \in S_1 \land A_i \text{ is a zero order name} \} \]
   and let Q be the subset of the S₁ such that
   \[ Q = \{ B_i \mid B_i \in S_1 \land B_i \notin P \}, \]
   define,

   **case I.** \( \exists \rho_c \in c \)
   \[ \forall \rho_d \in d \exists A_i \exists \rho_c[A_i] \neq \rho_d[A_i] \text{ then,} \]
   \[ c \cup d = \{ \rho_r \mid \rho_r = \rho_c \} \]

   **case II.** \( \exists \rho_d \in d \)
   \[ \forall \rho_c \in c \exists A_i \exists \rho_d[A_i] \neq \rho_c[A_i] \text{ then,} \]
   \[ c \cup d = \{ \rho_r \mid \rho_r = \rho_d \} \]

   **case III.** \( \exists \rho_c \in c \text{ and } \exists \rho_d \in d \)
   \[ \forall A_i \rho_d[A_i] = \rho_c[A_i] \text{ then,} \]
   \[ c \cup d = \{ \rho_r \mid \forall A_i \rho_r[A_i] = \rho_c[A_i] \land \forall B_i \rho_r[B_i] = \rho_c[B_i] \cup \rho_d[B_i] \} \]
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.

<table>
<thead>
<tr>
<th>Sem Yr</th>
<th>Dept</th>
<th>Crse#</th>
<th>#</th>
<th>Instructor</th>
<th>Student</th>
<th>Student Name</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall 89</td>
<td>CS</td>
<td>441</td>
<td>101</td>
<td>Fuller, J</td>
<td>042</td>
<td>Hu, Julia</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>109</td>
<td>Salona, Shyam</td>
<td>048</td>
<td>Hayte, Anne</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>451</td>
<td>104</td>
<td>Halem, Milt</td>
<td>052</td>
<td>Smith, P</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>110</td>
<td>Golf, Roger</td>
<td>049</td>
<td>Hite, Lee</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>463</td>
<td>102</td>
<td>Gordon, Chris</td>
<td>112</td>
<td>Virker, R</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>099</td>
<td>Oza, Amish</td>
<td>052</td>
<td>Smith, P</td>
<td>A</td>
</tr>
<tr>
<td>CS</td>
<td>451</td>
<td>104</td>
<td>Halem, Milt</td>
<td>110</td>
<td>Golf, Roger</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>112</td>
<td>Virker, R</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>453</td>
<td>102</td>
<td>Gordon, Chris</td>
<td>049</td>
<td>Hite, Lee</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>115</td>
<td>Selgam, R</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>098</td>
<td>Oza, Amish</td>
<td>052</td>
<td>Smith, P</td>
<td>A</td>
</tr>
<tr>
<td>Fall 90</td>
<td>CS</td>
<td>444</td>
<td>111</td>
<td>Wakim, N</td>
<td>032</td>
<td>Hu, Steve</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>110</td>
<td>Hwang, Steve</td>
<td>155</td>
<td>Waite, Terry</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>445</td>
<td>109</td>
<td>Salona, Shyam</td>
<td>113</td>
<td>Taylor, R</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>150</td>
<td>Wang, Paul</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>032</td>
<td>Berbert, John</td>
<td>052</td>
<td>Smith, P</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>441</td>
<td>101</td>
<td>Fuller, J</td>
<td>100</td>
<td>Faripati, P</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>104</td>
<td>Lee, Kyun-He</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>116</td>
<td>Shah, S</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>118</td>
<td>Tydings, Will</td>
<td>033</td>
<td>Henry, J</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>035</td>
<td>Elden, Tom</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>114</td>
<td>Taylor, Lisa</td>
<td>032</td>
<td>Berbert, John</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>052</td>
<td>Smith, P</td>
<td>052</td>
<td>Smith, P</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td>Faripati, P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>084</td>
<td>Lee, Kyun-He</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>116</td>
<td>Shah, S</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Spring 90</td>
<td>CS</td>
<td>457</td>
<td>109</td>
<td>Salona, Shyam</td>
<td>118</td>
<td>Tydings, Will</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 $(S_1 - S_2) \cap S_1$ such that $S_1 = S_2'$.

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

   \[ \exists \rho_c \in c \, \exists \rho_d \in d \, \forall A_i \, \rho_c[A_i] = \rho_d[A_i] \] and
   \[ \forall B_i \, \rho_c[B_i] \cap \rho_d[B_i] \neq \varnothing \] then,
   \[ c \cap d = \{ \rho_r \mid \forall A_i \, \rho_r[A_i] = \rho_c[A_i] \land \forall B_i \, \rho_r[B_i] = \rho_c[B_i] \cap \rho_d[B_i] \} \]

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.
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 $(S_1 - S_2) \cap S_1$ such that $S_1 = S_2'$.

2. Let $P$ be the subset of the $S_1$ such that

   \[ P = \{ A_i \mid A_i \in S_1 \land A_i \text{ is a zero order name} \} \]

   and let $Q$ be the subset of the $S_1$ such that

   \[ Q = \{ B_i \mid B_i \in S_1 \land B_i \notin P \} \]

   define,

Result of difference operation is generated as follows:

Case I. \( \exists \; \rho_c \in c \; \exists \; \rho_d \in d \) \( \forall \; A_i \; \rho_c[A_i] = \rho_d[A_i] \)

then,

\[ c - d = \{ \rho_r \mid \forall \; A_i \; \rho_r[A_i] = \rho_c[A_i] \land \forall \; B_i \; \rho_r[B_i] = \rho_c[B_i] - \rho_d[B_i] \} \]
Case II. \exists \rho_c \in c \forall \rho_d \in 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 \( \text{rcis}_1 \) and \( \text{rcis}_2 \) clusters, \( \text{rcis}_\text{diff} = \text{rcis}_1 \cdot \text{rcis}_2 \), is a cluster shown in Figure 3-10.

<table>
<thead>
<tr>
<th>Sem</th>
<th>Yr</th>
<th>Dept</th>
<th>Crse#</th>
<th>Id#</th>
<th>Instructor</th>
<th>Studio</th>
<th>Student</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall 89</td>
<td>CS</td>
<td>441</td>
<td>101</td>
<td>Fuller, J</td>
<td>041</td>
<td>Hu, Judy</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>442</td>
<td>103</td>
<td>Green, Jim</td>
<td>110</td>
<td>Gold, Roger</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>443</td>
<td>102</td>
<td>Gordon, Chris</td>
<td>112</td>
<td>Virkar, R</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>444</td>
<td>111</td>
<td></td>
<td>049</td>
<td>Hite, Lee</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>457</td>
<td>109</td>
<td>Solana, Shyam</td>
<td>114</td>
<td>Taylor, Lisa</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>458</td>
<td>101</td>
<td>Fuller, J</td>
<td>094</td>
<td>Lee, Kyyun-He</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>116</td>
<td>Shah, S</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>118</td>
<td>Tydings, Will</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-10: \( \text{rcis}_\text{diff} \) 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, $S_1 \cap S_2 = \emptyset$. The cross product of the two clusters, $c \times d$, is defined by

$$c[S_1] \times d[S_2] = r[S']$$

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 $S' = (\text{expand}(S_1), \text{expand}(S_2))$
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 $A_i \in S_1$ $\rho_r[A_i] = \rho_c[A_i]$

If $A_i \in S_2$ $\rho_r[A_i] = \rho_d[A_i]$

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, \( r[S'] = c \Join d \), is defined as follows:

Let

\[
X = \{ R_i \mid R_i \in S_1 \land R_i \text{ is a zero order name} \}
\]

Let

\[
Y = \{ R_i \mid R_i \in S_2 \land R_i \text{ is a zero order name} \}
\]

Let \( Z = X \cap Y \) then,

\[
S' = (X, Z, Y)
\]

\( r \) is derived from \( c' \) and \( d' \) as follows:

\[
r' = \{ \rho_r \mid \exists \rho_c \in c', \rho_d \in d' \land \forall z, \exists x \rho_c[Z] = \rho_d[Z]; \rho_r[z] = \rho_c[X] \land \rho_r[Y - Z] = \rho_d[Y - Z] \}
\]

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

<table>
<thead>
<tr>
<th>Sem</th>
<th>Yr</th>
<th>Dept</th>
<th>Gro#</th>
<th>Id#</th>
<th>Instructor</th>
<th>Studid</th>
<th>Stusname</th>
<th>Gr</th>
<th>Date_of_Birth</th>
<th>Date_Old</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall</td>
<td>89</td>
<td>CS</td>
<td>441</td>
<td>101</td>
<td>Fuller</td>
<td>041</td>
<td>Hsu, Judy</td>
<td>A</td>
<td>Jun 22, 1964</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Fall</td>
<td>89</td>
<td>CS</td>
<td>441</td>
<td>101</td>
<td>Fuller</td>
<td>049</td>
<td>Hsu, Judy</td>
<td>B</td>
<td>Feb 28, 1965</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Fall</td>
<td>89</td>
<td>CS</td>
<td>441</td>
<td>101</td>
<td>Fuller</td>
<td>052</td>
<td>Smith, P</td>
<td>A</td>
<td>Jan 04, 1965</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Fall</td>
<td>89</td>
<td>CS</td>
<td>441</td>
<td>109</td>
<td>Saloma, S</td>
<td>041</td>
<td>Hsu, Judy</td>
<td>A</td>
<td>Jun 22, 1964</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Fall</td>
<td>89</td>
<td>CS</td>
<td>441</td>
<td>109</td>
<td>Saloma, S</td>
<td>049</td>
<td>Hsu, Judy</td>
<td>B</td>
<td>Feb 28, 1965</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Fall</td>
<td>89</td>
<td>CS</td>
<td>441</td>
<td>109</td>
<td>Saloma, S</td>
<td>052</td>
<td>Smith, P</td>
<td>A</td>
<td>Jan 04, 1965</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Fall</td>
<td>89</td>
<td>CS</td>
<td>441</td>
<td>109</td>
<td>Saloma, S</td>
<td>052</td>
<td>Smith, P</td>
<td>A</td>
<td>Jan 04, 1965</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Fall</td>
<td>89</td>
<td>CS</td>
<td>441</td>
<td>109</td>
<td>Green, J</td>
<td>109</td>
<td>Goff, Roger</td>
<td>B</td>
<td>Aug 20, 1965</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Fall</td>
<td>89</td>
<td>CS</td>
<td>441</td>
<td>109</td>
<td>Green, J</td>
<td>110</td>
<td>Goff, Roger</td>
<td>B</td>
<td>Oct 02, 1959</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Fall</td>
<td>89</td>
<td>CS</td>
<td>442</td>
<td>103</td>
<td>Gordon, C</td>
<td>099</td>
<td>Goa, Anish</td>
<td>A</td>
<td>Nov 23, 1964</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Fall</td>
<td>89</td>
<td>CS</td>
<td>443</td>
<td>102</td>
<td>Gordon, C</td>
<td>049</td>
<td>Hsu, Judy</td>
<td>B</td>
<td>Feb 28, 1965</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Fall</td>
<td>89</td>
<td>CS</td>
<td>443</td>
<td>102</td>
<td>Gordon, C</td>
<td>099</td>
<td>Goa, Anish</td>
<td>A</td>
<td>Oct 02, 1959</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Spring</td>
<td>89</td>
<td>CS</td>
<td>457</td>
<td>109</td>
<td>Saloma, S</td>
<td>150</td>
<td>Wang, Paul</td>
<td>A</td>
<td>Jul 14, 1965</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Spring</td>
<td>89</td>
<td>CS</td>
<td>457</td>
<td>109</td>
<td>Saloma, S</td>
<td>052</td>
<td>Smith, P</td>
<td>A</td>
<td>Jan 04, 1965</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Spring</td>
<td>89</td>
<td>CS</td>
<td>458</td>
<td>101</td>
<td>Fuller, J</td>
<td>100</td>
<td>Pari, P</td>
<td>A</td>
<td>Dec 20, 1984</td>
<td>Sep 05, 1989</td>
</tr>
<tr>
<td>Spring</td>
<td>89</td>
<td>CS</td>
<td>458</td>
<td>101</td>
<td>Fuller, J</td>
<td>116</td>
<td>Shah, S</td>
<td>A</td>
<td>Apr 23, 1962</td>
<td>Sep 05, 1989</td>
</tr>
</tbody>
</table>

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


