BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 3

DCA2102

DATABASE MANAGEMENT SYSTEM

Unit 8

Relational Algebra

Table of Contents

SL No	Topic	Fig No / Table / Graph	SAQ / Activity	Page No
1	Introduction	-	-	3
	1.1 Objectives	-	-	
2	Basic Operations	1	1	4 - 8
	2.1 Union (∪)	2	-	
	2.2 Difference (-)	-	-	
	2.3 Intersection (∩)	-	-	
	2.4 Cartesian Product (x)	3	-	
3	Additional Relational Algebraic Operations	-	2	9 - 17
	3.1 Projection (π)	4	-	
	3.2 Selection (σ)	5	-	
	3.3 JOIN (⋈)	6, 7	-	
	3.4 Division (÷)	8	-	
4	Summary	-	-	18
5	Terminal Questions	-	-	18 - 19
6	Answers	-	-	19 - 20

1. INTRODUCTION

Relational algebras received little attention until the publication of E.F. Codd's relational model of data in 1970. Codd proposed such an algebra as a basis for database query languages. The first query language to be based on Codd's algebra was ISBL, and this pioneering work has been acclaimed by many authorities as having shown the way to make Codd's idea into a useful language. Even the query language of SQL is loosely based on a relational algebra, though the operands in SQL (tables) are not exactly relations and several useful theorems about relational algebra do not hold inthe SQL counterpart (arguably to the detriment of optimizers and/or users).

Relational algebra is a procedural language. It specifies the operations to beperformed on existing relations to derive result relations. Furthermore, it defines the complete scheme for each of the result relations. The relational algebraic operations can be divided into basic set-oriented operations and relational-oriented operations. The former are the traditional set operations, the latter, are those for performing joins, selection, projection, and division.

1.1 Objectives:

By the end of Unit 8, the learners should be able to understand:

- *Relational algebra and its history*
- *Basic operations like union, difference, and intersections*
- *Additional relational algebra operations including selection, project, joinand division operations.*

2. BASIC OPERATIONS

Basic operations are the traditional set operations: union, difference, intersection, and Cartesian product. Three of these four basic operations- union, intersection, and difference – require that operand relations be union compatible. Two relations are union compatible if they have the same parity and one-to-one correspondence of the attributes with the corresponding attributes defined over the same domain. The Cartesian product can be defined on any two relations. Two relations P and Q are said to be union compatible if both P and Q are of the same degree ***n*** and the domain of the corresponding ***n*** attributes are identical,

i.e.,

if P= { P1,..,Pn } and Q={ Q1,...Qn } then

Dom(P_i) = Dom(Q_i) for i = { 1,2...n }

where Dom(P_i) represents the domain of the attribute P_i.

Example 1

In the examples to follow, we utilize two relations P and Q given in Figure 8.1. R is a computed result relation. We assume that the relations P and Qin Figure 8.1 represent employees working on the development of software application packages J1 (say) and J2 (say) respectively.

P:

Id	**Name**
101	Jones
103	Smith
104	Lalonde
107	Evan
110	Drew
112	Smith

Q:

Id	**Name**
103	Smith
104	Lalonde
106	Byron
110	Drew

Fig 8.1: Union compatible relations

2.1 Union ($\cup$)

If we assume that P and Q are two union-compatible relations, then theunion of P and Q is set-theoretic union of P and Q.

The resultant relation, R = P U Q, has tuples drawn from P and Q such that

R = { t | ∈ P v t ∈ Q }

The result relation R contains tuples that are in either P or Q or in both of them. The duplicate tuples are eliminated.

Remember that from our definition of union compatibility the degree of the relations P and R is the same. The cardinality of the resultant relation depends on the duplication of tuples in P and Q.

From the above expression, we can see that if all the tuples in Q were contained in P, then |R| = |P| and R= P, while if the tuples in P and Q were disjoint, then |R| = |P|+|Q|.

Example 2

R, the union of P and Q given in Figure 8.1 in the above example 1 is shownin Figure 8.2(a). R represents employees working on the packages J_1 or J_2, or both of these packages. Since a relation does not have duplicate tuples, an employee working on both J_1 and J_2 will appear in the relation R only once.

R:

Id	Name
101	Jones
103	Smith
104	Lalonde
106	Byron
107	Evan
110	Drew
112	Smith

a) P $\cup$ Q

R:

Id	Name
101	Jones
107	Evan
112	Smith

b) P-Q

R:

Id	Name
103	Smith
104	Lalonde
110	Drew

c) P∩Q

Fig 8.2: Results of (a) union (b) difference and (c) intersection operations

2.2 Difference (-)

The difference operation removes common tuples from the first relation.

R = P-Q such that

$R = \{ t \mid t \in P \wedge t \not\in / Q \}$

Example 3

R, the result of P-Q, gives employees working only on package J1. (figure 8.2(b) in example 2). Employees working on both packages J1 and J2 have been removed.

2.3 Intersection (∩)

The intersection operation selects the common tuples from the two relations.

R = P ∩ Q where

$R = \{t \mid \in t P \wedge t \in Q\}$

Example 4

The resultant relation P∩Q is the set of all employees working on both the packages. (figure 8.2(c) of example 2).

The intersection operation is really unnecessary. It can be very simplyexpressed as:

P ∩ Q= P-(P-Q)

It is, however, more convenient to write an expression with a single intersection operation than one involving a pair of difference operations.

Note that in these examples the operand and the result relation schemes, including the attribute names, are identical i.e. P=Q=R. If the attribute names of compatible relations are not identical, the naming of the attributes of the result relation will have to be resolved.

2.4 Cartesian Product (x)

The extended Cartesian or simply the Cartesian product of two relations is the concatenation of tuples belonging to the two relations. A new resultant relation scheme is created consisting of all possible combinations of the tuples.

$$R = P \times Q$$

where a tuple $r \in R$ is given by $\{ t_1 || t_2 \mid t_1 \in P \wedge t_2 \in Q \}$, i.e. the result relation is obtained by concatenating each tuple in relation P with each tuplein relation Q. Here, we represent the concatenation operation.

The scheme of the result relation is given by:

$$R = P || Q$$

The degree of the result relation is given by:

$$|R| = |P| + |Q|$$

The cardinality of the result relation is given by:

$$|R| = |P|*|Q|$$

Example 5

The Cartesian product of the PERSONNEL relation and SOFTWARE_PACKAGE relations of figure 8.3(a) is shown in figure 8.3(b). Note that the relations P and Q from figure 8.1 of Example 1 are a subset of the PERSONNEL relation.

PERSONNEL:

Id	Name
101	Jones
103	Smith
104	Lalonde
107	Evan
110	Drew
112	Smith

Software Packages:

S
J_1
J_2

(a)

PERSONNEL:

Id	P. Name	S
101	Jones	J_1
101	Jones	J_2
103	Smith	J_1
103	Smith	J_2
104	Lalonde	J_1
104	Lalonde	J_2
106	Byron	J_1
106	Byron	J_2
107	Evan	J_1
107	Evan	J_2
110	Drew	J_1
110	Drew	J_2
112	Smith	J_1
112	Smith	J_2

(b)

Fig 8.3: (a) PERSONNEL (Emp#, Name) and SOFTWARE_PACKAGE(S) represent employees and software packages respectively; (b) the Cartesianproduct of PERSONNEL and SOFTWARE_PACKAGES.

The union and intersection operations are associative and commutative;therefore, given relations R, S, T:

$R \cup (S \cup T) = (R \cup S) \cup T = (S \cup R) \cup T = T \cup (S \cup R)=\ldots R \cap(S \cap T) =(R \cap S) \cap T=\ldots$

The difference operation, in general, is non-commutative and non-associative.

$R-S \neq S-R$ non-commutative

$R-(S-T) \neq (R-S)-T$ non-associative

Self-Assessment Questions – 1

1. In Relational Algebra the Basic operations are the traditional ________ orations.
2. The Cartesian product can be defined on any______relations.
3. If we assume that P and Q are two union-compatible relations, then theunion of P and Q is set-theoretic_______of P and Q.
4. The difference operation removes__________tuples from the first relation.
5. The_______operation selects the common tuples from the two relations.
6. The Cartesian product of two relations is the_________of tuplesbelonging to the two relations.
7. The union and intersection operations are associative and_______.
8. The__________operation, in general, is non-commutative and non-associative.

3. ADDITIONAL RELATIONAL ALGEBRAIC OPERATIONS

The basic set operations, which provide a very limited data manipulation facility, have been supplemented by the definition of the following operations: **projection, selection, join,** and **division**. These operations arerepresented by symbols **π**, **σ**, ⋈ and ÷ respectively. Projection and selectionare unary operations; join and divisions are binary.

3.1 Projection (π)

The projection of a relation is defined as a projection of all its tuples over some set of attributes, i.e., it yields a **vertical subset** of the relation. The projection operation is used to either reduce the number of attributes in the resultant relation or to reorder attributes. In the first case, the arity (or degree) of the relation is reduced. The projection operation is shown graphically in figure 8.4. Figure 8.4 shows the projection of the relation PERSONNEL on the attribute Name. The cardinality of the result relation is also reduced due to the deletion of duplicate tuples.

Id	**Name**	**Name**
101	Jones	Jones
103	Smith	Smith
104	Lalonde	Lalonde
106	Byron	Byron
107	Evan	Evan
110	Drew	Drew
112	Smith	Smith

Fig 8.4: Projection of relation PERSONNEL over attribute Name

We defined the projection of a tuple ti over the attribute A, denoted $t_i[A]$ or $\pi A(t_i)$, as (a), where a is the value of tuple t_i over the attribute A. Similarly, we define the projection of a relation T, denoted by T[A] or $\pi A(t_i)$, on the attribute A. This is defined in terms of the projection for each tuple in ti belonging to T on the attribute A as:

$$T[A] = \{ a_i \mid t_i [A] = a_i \ t_i \in T\}$$

where T[A] is a single attribute relation and | T[A] | £ T. The cardinality T[A] may be less than the cardinality |T| because of the deletion of any duplicates in the result. A case in point is illustrated in figure 8.4.

Similarly, we can define the projection of a relation on a set of attribute names, X, as a concatenation of the projections for each attribute A in X for every tuple in the relation.

$T[X] = \{ t_i[A] \mid t_i \in T \}$

Here A belongs to X where $t_i[A]$ represents the concatenation of all $t_i[A]$ for all $A \in X$ (A belongs to X)

Simply stated, the projection of a relation P on the set of attribute names Y belongs to P is the projection of each tuple of the relation P on the set of attribute names Y.

Note that the projection operation reduces the arity if the number of attributes in X is less than the arity of the relation. The projection operation may also reduce the cardinality of the result relation since duplicate tuples are removed. (Note that the projection operation produces a relation as the result. By definition, a relation cannot have duplicate tuples. In most commercial implementations of the relational model, however, the duplicates would still be present in the result).

3.2 Selection (σ)

Suppose we want to find those employees in the relation PERSONNEL of figure 8.3(a) of Example 5 with an Id less than 105. This is an operation that selects only some of the tuples in the relation. Such an operation is known as a **selection operation**. The projection operation yields a vertical subset of arelation. The action is defined over a subset of the attribute names but overall the tuples in the relation. The action is defined over a subset of the attribute names but overall the tuples in the relation. The selection operation, however, yields a horizontal subset of a given relation, i.e., the action is defined over a complete set of attribute names but only a subset of

the tuples are included in the result. To have a tuple included in the result relation, the specified selection conditions or predicates must be satisfied byit. The selection operation is sometimes known as the **restriction operation**.

PERSONNEL:

Id	Name
101	Jones
103	Smith
104	Lalonde
107	Evan
110	Drew
112	Smith

Results of Selection

Id	Name
101	Jones
103	Smith
104	Lalonde

Fig 8.5: Result of Selection over PERSONNEL for Id< 105

Any finite number of predicates connected by Boolean operators may be specified in the selection operation. The predicates may define a comparison between two domain-compatible attributes or between an attribute and a constant value; if the comparison is between attribute A1 andconstant c1, then c1 belongs to **Dom (A1)**.

Given a relation P and a predicate expression B, the selections of those tuples of relation P that satisfy the predicate B is a relation R written as:

$R = \sigma B (P)$

The above expression could be read as "select those tuples t from P in which the predicate B(t) is true". The set of tuples with R is in this case defined as follows:

$R= \{ t \mid t \in \text{to } P \;\; B (t) \}$

3.3 JOIN (⋈)

The join operator, as the name suggests, allows the combining of two relations to form a single new relation. The tuples from the operand relations that participate in the operation and contribute to the result are related. The join operation allows the processing of relationships existing between the operand relations.

Example 6

In figure 8.6 we encounter the following relations:

ASSIGNMENT (Emp#, Prod#, Job#)

JOB_FUNCTION (Job#, Title)

EMPLOYEE:

Emp#	Name	Profession
101	Jones	Analyst
103	Smith	Programmer
104	Lalonde	Receptionist
106	Byron	Receptionist
107	Evan	VPR & D
110	Drew	VP operations
112	Smith	Manager

PRODUCT:

Prod#	Prod-Name	Prod-details
HEAP1	HEAP-SORT	ISS Module
BINS9	BINARY-SEARCH	ISS/R Module
FM6	FILE-MANAGER	ISS/R-PC Subsys
B++1	B++_TREE	ISS/R Turbo Sys
B++2	B++_TREE	VPR & D

a)

JOB-FUNCTION:

Job#	Title
1000	CEO
900	President
800	Manager
700	Chief Programmer
600	Analyst

ASSIGNMENT

Emp#	Prod#	Job#
107	HEAP1	800
101	HEAP1	600
110	BINS9	800
103	HEAP1	700
101	BINS9	700
110	FM6	800
107	B++1	800

b)

Fig 8.6: (a) Relation schemes for employee role in development teams

b) Sample relations

Suppose we want to respond to the query "Get product number of assignments whose development teams have a **chief programmer".** This requires first computing the Cartesian product of the ASSIGNMENT and JOB_FUNCTION relations. Let us name this product relation TEMP. This is followed by selecting those tuples of TEMP where the attribute Title has the value chief programmer and the value of the attribute Job# in ASSIGNMENT and JOB_FUNCTION are the same. The required result, shown below is obtained by projecting these tuples on the attribute Prod#. The operations are specified below.

TEMP = (ASSIGNMENT X JOB_FUNCTION)

$\pi_{Prod\#}$(σ Title = 'chief programmer' ∪ ASSIGNMENT.Job# (TEMP))

Prod#
HEAP1
BINS9

In another method of responding to this query, we can first select those tuples from the JOB_FUNCTION relation so that the value of the attribute Title is a chief programmer. Let us call this set of tuples the relation TEMP1. We then compute the Cartesian product of TEMP1 and ASSIGNMENT, calling the product TEMP2. This is followed by a projection on Prod# over TEMP2 to give us the required response. These operations are specified below:

TEMP1 = (σ Title = 'chief programmer' (JOB_FUNCTION))

TEMP2 = (σ ASSIGNMENT.Job# = JOB_FUNCTION.Job# (ASSIGNMENTX TEMP1))

$\pi_{Prod\#}$ (TEMP2) gives the required result.

Notice that in the selection operation that follows the Cartesian product we take only those tuples where the value of the attributes ASSIGNMENT.Job#and JOB_FUNCTION.Job# are the same. These combined operations of the cartesian product followed by selection are the join operation. Note that we have qualified the identically named attributes by the name of the corresponding relation to distinguishing them.

In case of the join of a relation with itself, we would need to rename either the attributes of one of the copies of the relation or the relation name itself. We illustrate this in example 7.

In general, the join condition may have more than one term, necessitating the use of the subscript in the comparison operator. Now we shall define the different types of join operations.

In these discussions, we use P, Q, R, and so on to represent both the relation scheme and the collection or bag of underlying domains of the attributes. We call it a bag of domains because more than one attribute may be defined on the same domain.

Typically, P **C** Q may be null and this guarantees the uniqueness of attributenames in the result relation. When the same attribute name occurs in thetwo schemes we use qualified names.

Two common and very useful variants of the join are the equi-join and the natural join. In the **equi-join** the comparison operator theta (i=1,2,...,n) is always the equality operator (=). Similarly, in the natural join, the comparison operator is always the equality operator. However, only one of the two sets of domain compatible attributes involved in the natural join is $\mathbf{A_i}$ from P and $\mathbf{B_i}$ from Q, for i = 1,...n, the natural join predicate is a conjunction of terms of the following form:

$$(t1[A_i] = t2[B_i]) \text{ for } i = 1,2...n$$

Domain compatibility requires that the domains of A_i and B_i be compatible, and for this reason relation scheme P and Q have attributes defined on common domains, i.e., P **C** Q^1. Therefore, join attributes have common domains in the relation schemes P and Q. Consequently, only one set of thejoin attributes on these common domains needs to be preserved in theresult relation. This is achieved by taking a projection after the join operation, thereby eliminating the duplicate attributes. If the relations P and Q have attributes with the same domain, but different attribute names, then renaming or projection may be specified.

Example 7

Given the EMPLOYEE and SALARY relations of figure 8.7(i), if we have to find the salary of employees by name, we join the tuples in the relation EMPLOYEE with those in SALARY such that the value of the attribute Id in EMPLOYEE is the same as that in SALARY. The natural join takes the predicate expression to be EMPLOYEE.Id = SALARY.Id. The result of the natural join is shown in figure 8.7(ii). When using the natural join, we do not need to specify this predicate. The expression to specify the operation of finding the salary of employees by name is given as follows. Here we project the result of the natural join operation on the attributes Name and Salary:

π(Name. Salary)(EMPLOYEE |x| SALARY)

EMPLOYEE:

Id	Name
101	Jones
103	Smith
104	Lalonde
107	Evan

SALARY:

Id	Salary
101	67
103	55
104	75
107	80

EMPLOYEE |X| SALARY

Id	Name	Salary
101	Jones	67
103	Smith	55
104	Lalonde	75
107	Evan	80

ASSIGNMENT.Emp#	COASSIGN.Emp#
107	107
107	101
107	103
101	107
101	101
101	103
110	101
103	107
103	101
103	103
101	110

Fig 8.7: i) The natural join of EMPLOYEE and SALARY relations;
ii) The joint of ASSIGNMENT with the renamed copy

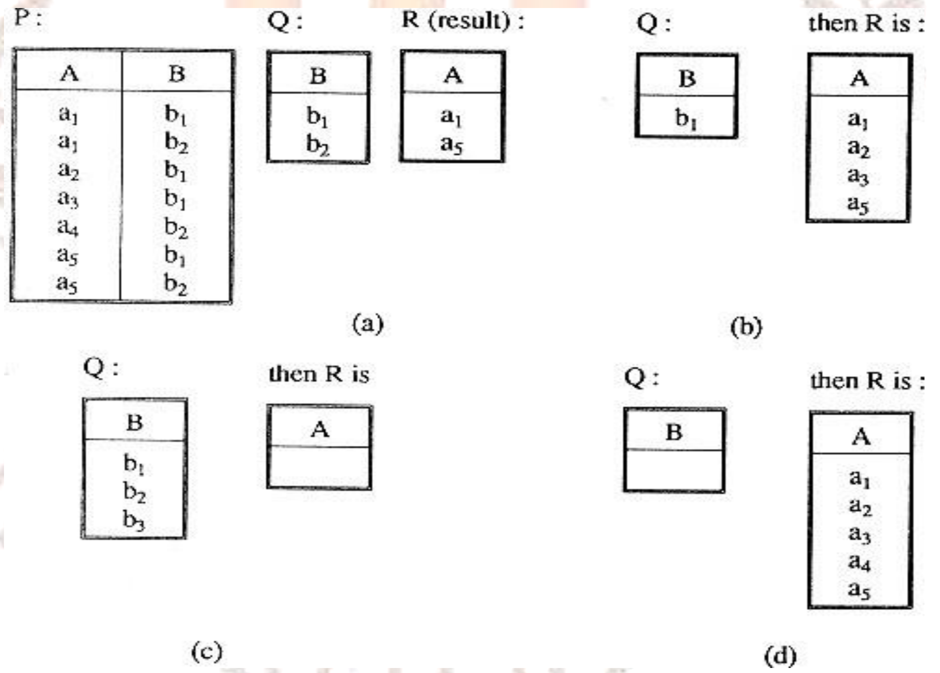
3.4 Division (÷)

Before we define the division operation, let us consider an example.

Example 8

Given the relations P and Q, as shown in figure 8.8(a), the result of dividingP by Q is the relation R and it has two tuples. For each tuple in R, its product with the tuples of Q must be in P. In our example (a_1,b_1) and (a_2,b_2) must both be tuples in P; the same is true for (a_5,b_1) and (a_5,b_2).

Simply stated, the cartesian product of Q and R is a subset of P. In figure 8.8(b), the result relation R has four tuples; the cartesian product of R and Qgives a resulting relation which is again a subset of P. In figure 8.8(c) since there are no tuples in P with a value b_3 for the attribute B (i.e., selecting B=b_3(P) = 0), we have an empty relation R, which has a cardinality of zero.



P :

A	B
a_1	b_1
a_1	b_2
a_2	b_1
a_3	b_1
a_4	b_2
a_5	b_1
a_5	b_2

Q :

B
b_1
b_2

R (result) :

A
a_1
a_5

(a)

Q :

B
b_1

then R is :

A
a_1
a_2
a_3
a_5

(b)

Q :

B
b_1
b_2
b_3

then R is

A

(c)

Q :

B

then R is :

A
a_1
a_2
a_3
a_4
a_5

(d)

Fig 8.8: Examples of the division operation. (a) R = P / Q; (b) R = P / Q(P is same in part i); (c) R=P/Q (P is same as in part i); (d) R= P/Q (P is same as in part i)

In figure 8.8(d), the relation Q is empty. The result relation can be definedas the projection of P on the attributes in P-Q. However, it is usual to disallow division by an empty relation.

Finally, if relation P is an empty relation, then relation R is also an empty relation.

Let us treat the Q as representing one set of properties (the properties are defined on the Q, each tuple in Q representing an instance of these properties) and the **relation r** as representing entities with these properties (entities are defined on P-Q, and the properties are, as before, defined on Q); note that P ∪ Q must be equal to P. Each tuple in P represents an object with some given property. The resultant relation R, then, is the set of entities that possesses all the properties specified in Q. The two entities a_1 and a_5 possess all the properties, i.e., b_1 and b_2. The other entities in P, a_2, a_3, and a_4, only possess one, not both, of the properties. The divisionoperation is useful when a query involves the phrase "for all objects having all the specified properties." Note that both P-Q and Q in general representa set of attributes. It should be clear that Q is not a subset of P.

Self-Assessment Questions – 2

9. The projection of a relation yields a____________subset of the relation.
10. The projection operation may________the cardinality of the resultrelation since duplicate tuples are removed.
11. The selection operation yields a___________subset of a given relation.
12. The___________operator allows the combining of two relations toform a single new relation.
13. The join operation allows the processing of____________existing between the operand relations.
14. Two common and very useful variants of the join are the equi-join and the _________.
15. In the ______________ the comparison operator theta (i=1,2,...,n) is always the equality operator (=).
16. In the natural join the ______________ operator is always the equality operator.

4. SUMMARY

In this unit, we discussed that relational algebra defines a set of algebraic operations that operate on tables, and output tables as their results. These operations can be combined to get expressions that express desired queries. The algebra defines the basic operations used within relational query languages.

The operations in relational algebra can be divided into two types. One is Basic operations and the other is Additional operations that can be expressed interms of the basic operations. We used relational algebra with the assignment operator to express these modifications.

5. TERMINAL QUESTIONS

1. Explain the statement that relational algebra operators can be *composed*. Why is the ability to compose operators importantly?
2. Given two relations $R1$ and $R2$, where $R1$ contains N1 tuples, $R2$ contains N2 tuples, and N2 > N1 > 0, give the minimum and maximum possible sizes (in tuples) for the result relation produced by each of the following relational algebra expressions. In each case, state any assumptions about the schemas for $R1$ and $R2$ that are needed to makethe expression meaningful:
   a. $R1 \cup R2$
   b. $R1 \cap R2$
   c. $R1 - R2$
   d. $R1 \times R2$
   e. $\sigma_{a=5}$ (R1)
   f. π_a (R1)
   g. $R1/R2$
3. Consider the following schema:

   Suppliers(*sid:* integer, *sname:* string, *address:* string)

   Parts(*pid:* integer, *pname:* string, *color:* string)

   Catalog(*sid:* integer, *pid:* integer, *cost:* real)

The key fields are underlined, and the domain of each field is listed after thefield name. Thus *sid* is the key for Suppliers, *pid* is the key for Parts, and *sid*and *pid* together form the key for Catalog. The Catalog relation lists the prices charged for parts by Suppliers. Write the following queries in relational algebra.

1. Find the *names* of suppliers who supply some red part.
2. Find the *sids* of suppliers who supply some red or green part.
3. Find the *sids* of suppliers who supply some red part or are at 221Packer Ave.
4. Find the *sids* of suppliers who supply some red part and some greenpart.
5. Find the *sids* of suppliers who supply every part.
6. Find the *sids* of suppliers who supply every red part.
7. Find the *sids* of suppliers who supply every red or green part.
8. Find the *sids* of suppliers who supply every red part or supply everygreen part.

6. ANSWERS

Self-Assessment Questions

1. Set
2. Two
3. Union
4. Common
5. Intersection
6. Concatenation
7. Commutative Difference
8. Vertical
9. Reduce
10. Horizontal
11. Join
12. Relationships
13. natural join
14. equi-join
15. comparison

Terminal Questions

1. A set of basic operators in the Relational Algebra can take relations as their operands and return the result as a relation. Hence the output of one operation may be used as input to another operation. In other words operators are composed. (Refer section 3 for detail)
2. Refer section 2 for detail.
3. Refer section 2 and 3 for detail.