Relational Algebra

Relational algebra is an algebra which manipulates tables to produce new tables. The word "relational" is derived from pure mathematics. Using this terminology tables are **Relations**, columns are **Attributes** and rows are **Tuples**.

In relational algebra tables are treated as sets of rows and the usual rules about sets apply. So relations never contain duplicate rows and the order of rows within a table is irrelevant.

The most important operators of this algebra are Select, Project and Join. The **Select** operator picks out specific rows from a table to make a new table with the same columns as the original but possibly fewer rows. It selects which rows go into the new table on the basis of values contained in the row. The syntax of this operator is

SELECT <boolean expression> **(** <relation identifier> **)**

Text in bold is actually typed; <boolean expression> and <relation identifier> are both replaced by the things named between the < and >. The expression produces a table which contains those rows from the table described within the brackets for which the boolean expression (predicate) is true.

Given a relation called **Letters&Numbers**

Letters	Numbers
A	1
B	2
C	3
C	1

SELECT Letters=C (Letters&Numbers) is

Letters	Numbers
C	3
C	1

The **Project** operator picks out columns from a table to make a new table. The new table produced never contains duplicate rows because no relation can ever contain duplicate tuples. So the project operator picks out some of the columns from a table and then removes any duplicate rows to produce the new table. It does not matter which rows are eliminated because the order of rows is irrelevant. The syntax of the Project operator is

PROJECT <column list> **(** <relation identifier> **)**

where the <column list> is a list of the column names separated by commas. The <relation identifier> need not just be a table name; it could be another relational algebra expression. This is also true for Select above.

PROJECT Numbers (Letters&Numbers) is

Numbers
1
2
3

The **Join** operator produces a new table from two other tables. It does this by concatenating pairs of rows, one from each of the original tables, to make a new row. Only some of the possible pairs of rows are joined. It will only concatenate two tables if they have a pair of columns, one in each table, which

have the same domain. Individual pairs of rows are concatenated if they contain the same value in each of the two matching columns. The Join operator produces a table with all the attributes of both the original tables but the column the two tables are joined on is not repeated as it would be if the new tuples were produced by simply running the two existing tuples together. This is known as the **natural join**. The syntax of the join operator is

<relation identifier> **JOIN** <relation identifier>

In this case too the relation identifiers can either be relation names or other relational algebra expressions.

Given two relations **Letters&Numbers** and **NumbersSymbols&Brackets**

Letters	Numbers
A	1
B	2
C	3
C	1

Numbers	Symbols	Brackets
1	/	{
1	*	(
2	*	[
4	*	(

Letters&Numbers JOIN NumbersSymbols&Brackets is

Letters	Numbers	Symbols	Brackets
A	1	/	{
A	1	*	(
B	2	*	[
C	1	/	{
C	1	*	(

Join also works if there is more than one common column; the two tables are joined on any columns they have in common. If the tables have more than one column in common then pairs of rows are only joined if all the common column values match.

Since relations are effectively sets of rows, we can use the standard set operators **Union**, **Intersection** and **Difference** in relational algebra too. All three operators take two tables and produce a new one. The two original tables must be the same type; which means they must have the same number of columns with the same domains in the same order. The resulting table is the same type as the two originals.

Their syntax is

<relational identifier> **UNION** <relational identifier>

which produces a table with all the rows of both tables but no duplicates,

<relational identifier> **INTERSECTION** <relational identifier>

which produces a table with only those rows which occur in both tables and

<relational identifier> **DIFFERENCE** <relational identifier>

which produces a table with all the rows which occur in the first table but not the second.

PROJECT Numbers (Letters&Numbers) DIFFERENCE
PROJECT Numbers (NumbersSymbols&Brackets)

Numbers
3

PROJECT Numbers (Letters&Numbers) INTERSECTION
PROJECT Numbers (NumbersSymbols&Brackets)

Numbers
1
2