Relational Algebra

Relational algebra is another way of expressing queries. Each expression specifies a relation (table). It is **NOT** implemented on machines.

1. `Select function.` The select function selects rows in the plugged in table. Subscript predicates specify which rows. Note this is like the where-clause in SQL. It is NOT like the select-clause. The Greek letter lower case sigma is the symbol for the function. Example the query

$$\sigma_{can\_id=8}(Contribution)$$

is all rows of Contribution where the candidate id is 8. The predicates in the subscript can be connected with and's and or's.

2. `Project function.` The project function selects columns in the plugged in table. Subscripts specify which columns. This is like the select-clause in SQL. The Greek letter pi is the symbol for the project function. Example the query

$$\Pi_{amt,date}(Contribution)$$

makes a table consisting of two columns, `amt, date`, that contains all the amt and date data from the Contribution table.

3. `Natural join operation.` The "bowtie" below is the symbol for the natural join operation. Example:

$$Candidate \bowtie Contribution$$

is the natural join of Candidate and Contribution. The natural join here is just like the SQL natural join.

4. `Cross product operation.` The "cross" is below is the symbol for the cross operation. Example:

$$T1 \times T2$$

that works just like SQL cross join operation.

5. `Set operations: Union, Intersection, Difference.` Each table is considered to be a set of (its) rows. If two tables have all the same columns, then we can perform the union, intersect and diffence operations on them. Suppose that tables T1 and T4 each have columns A and B. If that tables T1 and T4 are as shown below:

table T1:

A	B
1	3
2	4

table T4:

A	B
1	7
3	4
1	3

Then results of union, intersection and difference is:

$T1 \cup T4 =$

A	B
1	3
2	4
1	7
3	4

$T1 \cap T4 =$

A	B
1	3

$T1 - T4 =$

A	B
2	4

6. `Aggregation operator.` The columns of the result table are the subscripts left and right of the symbol for aggregation, the script G you see below. The subscripts to the right are the aggregate functions applied to the attributes of the plugged-in table. The subscripts to the left are attributes of the plugged-in table to be grouped on. Example:

$$_{can\_id}\mathcal{G}_{count(distinct\ contr\_id)}(Contribution)$$

The result table has two columns: can_id and count(distinct contr_id). For each value of can_id the group of all contributions with that value is formed. Then the number of distinct contr_id's is computed (counted). That is the number that goes into the second column. This aggregates the rows of Contribution into groups have the same can_id. And for each group counts the number of distinct contr_id in the rows in the group. The aggregate functions we use in SQL (count, sum, max, avg, etc.) can be used in relational algebra. Another example:

$$\mathcal{G}_{avg(amt)}(Contribution)$$

Here the only group is the whole table, Contribution. This relational algebra finds the average contribution made over the whole set of contributions.