A new keyword: `(... GROUP BY <attr>)`

Full structure of RQL

By adding a `GROUP BY` clause, we want to group the result set by one or more column conjugated by aggregation functions. The full structure of an RQL query expression now becomes:

```
( SELECT <attrs>
  FROM <tables>
  WHERE <condition>
  GROUP BY <attr>
  ORDER BY <order> )
```

Syntaxes of `GROUP BY`:

```
(define Employee
  '(("eid" "name" "salary" "dept")
    (1 "Jack" 100 "A")
    (2 "John" 150 "A")
    (3 "Tom" 250 "B")
    (4 "Alice" 180 "B")))
(define Department
  '(("did" "name")
    (1 "A")
    (2 "B")))
```

Here are two sample queries with `GROUP BY`:

- We specify aggregation function in `SELECT` with `["new-column-name" (aggregation-func "attr-name")]`. In below example, we want to aggregate `"salary"` column with the **predefined** function `sum` and rename the new column as `"sum"`.

```
> (SELECT '("dept" ["sum" (sum "salary")])
   FROM [Department "D"] [Employee "E"]
   WHERE (equals? "D.name" "E.dept")
   GROUP BY "dept"
   ORDER BY "sum")
'(("dept" "sum")
  ("A" 250)
  ("B" 430))
```

- On the other hand, we want our `GROUP BY` to support multiple aggregation functions. (eg. `SUM`, `AVG`, `COUNT` in SQL). And we can have multiple aggregation functions in the same table. We can also specify many aggregation functions to get more than one columns. The new columns can be further used as `<order>` in `ORDER BY`.

```
> (SELECT '("dept" ["sum" (sum "salary")] ["max" (max "salary")]
   FROM [Department "D"] [Employee "E"]
   WHERE (equals? "D.name" "E.dept")
   GROUP BY "dept"
   ORDER BY "max")
'(("dept" "sum" "max")
  ("B" 430 250)
  ("A" 250 150))
```

Order of execution in RQL:

Before we add this new keyword `GROUP BY`, the order of execution in RQL is: `FROM -> WHERE -> ORDER BY -> SELECT` and now the new order is `FROM -> WHERE -> GROUP BY -> ORDER BY -> SELECT`.

First, we use `FROM` and `WHERE` clauses to get a temporary table. Then we use `GROUP BY` clauses to call the aggregation functions, and output a new table. After using the `ORDER BY` clause to sort our table, we can take out the columns we want by `SELECT`. **The reason why we put `GROUP BY` before `ORDER BY` and `SELECT` is that we want to add the functionality of ordering by the output of aggregation functions.**