Q LANGUAGE - FUNCTIONAL QUERIES

Functional *Dynamic* queries allow specifying column names as symbols to typical q-sql select/exec/delete columns. It comes very handy when we want to specify column names dynamically.

The functional forms are −

```
?[t;c;b;a]    / for select
![t;c;b;a]    / for update
```

where

- **t** is a table;
- **a** is a dictionary of aggregates;
- **b** the by-phrase; and
- **c** is a list of constraints.

Note −

- All **q** entities in **a**, **b**, and **c** must be referenced by name, meaning as symbols containing the entity names.
- The syntactic forms of select and update are parsed into their equivalent functional forms by the **q** interpreter, so there is no performance difference between the two forms.

Functional select

The following code block shows how to use **functional select** −

```
q)t:([]n:`ibm`msft`samsung`apple;p:40 38 45 54)

q)t

    n       p
--------------------
   ibm     40
   msft    38
 samsung   45
  apple    54

q)select m:max p,s:sum p by name:n from t where p>36, n in `ibm`msft`apple

  name |   m   s
------ | ---------
 apple |  54  54
 ibm   |  40  40
 msft  |  38  38
```

Example 1

Let’s start with the easiest case, the functional version of **“select from t”** will look like −

```
q)?[t;();0b;()]     / select from t

    n      p
-----------------
   ibm    40
   msft   38
```

```
 samsung  45
  apple   54
```

Example 2

In the following example, we use the enlist function to create singletons to ensure that appropriate entities are lists.

```
q)wherecon: enlist (>;`p;40)

q)?[`t;wherecon;0b;()] / select from t where p > 40

    n      p
----------------
 samsung  45
  apple   54
```

Example 3

```
q)groupby: enlist[`p] ! enlist `p

q)selcols: enlist [`n]!enlist `n

q)?[ `t;(); groupby;selcols]        / select n by p from t

   p  |    n
----- | -------
  38  |  msft
  40  |  ibm
  45  | samsung
  54  | apple
```

Functional Exec

The functional form of exec is a simplified form of **select**.

```
q)?[t;();();`n]                / exec n from t (functional form of exec)
`ibm`msft`samsung`apple

q)?[t;();`n;`p]                / exec p by n from t (functional exec)

apple   | 54
ibm     | 40
msft    | 38
samsung | 45
```

Functional Update

The functional form of update is completely analogous to that of **select**. In the following example, the use of enlist is to create singletons, to ensure that input entities are lists.

```
q)c:enlist (>;`p;0)

q)b: (enlist `n)!enlist `n

q)a: (enlist `p) ! enlist (max;`p)

q)![t;c;b;a]

   n      p
-------------
  ibm    40
  msft   38
 samsung 45
 apple   54
```

Functional delete

Functional delete is a simplified form of functional update. Its syntax is as follows −

```
![t;c;0b;a]        / t is a table, c is a list of where constraints, a is a
                    / list of column names
```

Let us now take an example to show how functional delete work −

```
q)![t; enlist (=;`p; 40); 0b;`symbol$()]
                                              / delete from t where p = 40
   n       p
---------------
  msft    38
 samsung  45
 apple    54
```