

Using Characteristic Functions

Characteristic functions are a mechanism for taking logic out of an SQL where-clause and putting it into an SQL select-list. It does this by converting boolean expressions into corresponding numeric expressions that can be used in calculations rather than in testing if something is true or false.

For example, suppose we want to compare $A = B$ where A and B are numeric. The expression $1 - \text{abs}(\text{sign}(A - B))$ has the property that its value is 1 iff $A = B$ and 0 iff $A \neq B$. The $\text{abs}()$ function is the well-known absolute value and $\text{sign}()$ is a function that returns 1, 0 or -1 depending on whether its argument is positive, zero or negative. There is a list of these functions, called incidentally characteristic functions, for all familiar boolean expressions.

| boolean expression | characteristic function |
|--------------------|---|
| $A = B$ | $1 - \text{abs}(\text{sign}(A - B))$ |
| $A < B$ | $1 - \text{sign}(1 + \text{sign}(A - B))$ |
| $A > B$ | $1 - \text{sign}(1 - \text{sign}(A - B))$ |
| $A \leq B$ | $\text{sign}(1 - \text{sign}(A - B))$ |
| $A \geq B$ | $\text{sign}(1 + \text{sign}(A - B))$ |
| $A \neq B$ | $\text{abs}(\text{sign}(A - B))$ |

We want to use these functions on SQL expressions but need to give them a name space that will not interfere with other functions and reserved words so we prefix all these function names with the letter q to come up with the list of user-defined functions in our previous document.

| user-defined function | meaning |
|-----------------------|---------------------------------------|
| $qeq(A, B)$ | returns 1 if $A == B$; 0 otherwise |
| $qlt(A, B)$ | returns 1 if $A < B$; 0 otherwise |
| $qgt(A, B)$ | returns 1 if $A > B$; 0 otherwise |
| $qle(A, B)$ | returns 1 if $A \leq B$; 0 otherwise |
| $qge(A, B)$ | returns 1 if $A \geq B$; 0 otherwise |
| $qne(A, B)$ | returns 1 if $A \neq B$; 0 otherwise |

In addition to these functions we can even add boolean logic functions such as $\text{and}()$, $\text{or}()$ and $\text{not}()$.

| boolean expression | characteristic function |
|--------------------|---|
| $qand(A, B)$ | $\text{sign}(\text{abs}(A * B))$ |
| $qor(A, B)$ | $\text{sign}(\text{abs}(A) + \text{abs}(B))$ |
| $qnot(A)$ | $\text{sign}(\text{abs}(1 - \text{abs}(\text{sign}(A))))$ |

Of course, these should only be used with arguments that are the results of other characteristic functions since they expect integer input.

We can now combine the numeric and boolean functions to create even more sophisticated logic.

| | |
|--------------------|------------------------------|
| boolean expression | characteristic function |
| $qbtn(A, B, C)$ | $qand(qge(a, b), qle(a, c))$ |

Now our challenge is to use them to produce results we otherwise could only achieve with a lot of complicated SQL or worse yet, by transferring the logic to the user-interface and writing it there in VB or Java, etc.

Table Pivoting

This is an example where we try to turn data found in table rows into data found in the result table columns. Suppose we want to produce a table whose schema is

| Publisher | Num Copies < \$10 | Num Copies between \$10 and \$20 | Num Copies over \$20 |
|-----------|-------------------|----------------------------------|----------------------|
|-----------|-------------------|----------------------------------|----------------------|

The SQL that does this is:

```
select pub_name , sum( qlt(p_price,10)) as LT10,
       sum( qbtn(p_price,10,20)) as BT10_AND_20,
       sum( qgt(p_price,20)) AS GT20
from book bk, copy c
where bk.isbn = c.isbn
group by pub_name
```

The output is

| PUB_NAME | LT10 | BT10_AND_20 | GT20 |
|----------|------|-------------|------|
| AW | 0 | 6 | 3 |
| CSP | 0 | 1 | 1 |
| PH | 0 | 0 | 4 |
| Wiley | 0 | 0 | 1 |

Medians:

A second type of calculation , statistical this time, might be to find all books that cost less than the median price. The SQL is

```
select author, title from book
where c_price <
( select b1.c_price from book b1, book b2
  group by b1.c_price
  having sum(qle(b2.c_price, b1.c_price)) = (count(*) + 1)/2);
```

The output is

| AUTHOR | TITLE |
|--------|-------|
| Ullman | DB |
| K'rock | Queue |
| Br'kes | MMM |

Histograms:

Imagine we want to find the number of books with 1, 2 or more than 2 copies in the library. The following SQL produces the information about how many copies there are of each book

```
select isbn, count(*) as cnt from copy group by isbn;
```

Next we want to use this info to count how many books there are with 1, 2 or more copies.

```
select sum(qeq(t.cnt,1)) as one, sum(qeq(t.cnt,2)) as two,
       sum(qgt(t.cnt,2)) as many
from table(select isbn, count(*) as cnt from copy group by
           isbn) as t;
```

The output is

| ONE | TWO | MANY |
|-----|-----|------|
| 3 | 2 | 2 |

Notice we have not *group by* in the outer query because we only have one row of output so only one group.

Exercises:

Answer the following queries.

1. Find the median current price for a book published by Addison-Wesley.
2. Find the number of cardholders who have borrowed one book, two books or more than two books.
3. Write a user-defined function that evaluates *and()* for three arguments.