SQL | Alternative Quote Operator

classyallrounder

This post is a continuation of the SQL Concatenation Operator.

Now, suppose we want to use **apostrophe** in our literal value but we can't use it directly.

See **Incorrect** code:
SELECT id, first_name, last_name, salary,
first_name||' has salary's '||salary
AS "new" FROM one

So above we are getting error, because Oracle server thinking that the **first apostrophe** is for the **starting literal** and the **second apostrophe** is for the **ending literal**, so what about the **third apostrophe**???. That's why we get error.

Alternative Quote Operator(*q*)

:
To **overcome** the above problem Oracle introduce an operator known as **Alternative Quote Operator**(q).

Let's see through an example:

```
Query that uses Alternative Quote Operator(q)
SELECT id, first_name, last_name, salary,
first_name||q'{ has salary's }'||salary
AS "new" FROM myTable
```

```
Output:
See, we are able to use apostrophe in the
new column as a literal value of myTable

ID  FIRST_NAME LAST_NAME  SALARY           new
3    Shane     Watson     50000    Shane has salary's 50000
1    Rajat     Rawat      10000    Rajat has salary's 10000
2    Geeks     ForGeeks   20000    Geeks has salary's 20000
3    MS        Dhoni      90000    MS    has salary's 90000
```

Here above see, **q'{** indicates starting of our literal value and then we use **}'** which indicates end of our literal value. So see here we have used apostrophe in our literal value easily(means we easily use **'s** in salary) without any error that's why we get output as Rajat has salary**'s** 50000.

So to use apostrophe in literal we first need to use **q** which is known as **alternative quote operator** after that we need to use an apostrophe ' and after that we need to use a **delimiter** and after delimiter we write our literal value, when we finished writing our literal value then again we need to **close the delimiter** which we have **opened before** and after that we need to put an **apostrophe again** and hence in this way we can use apostrophe in our literal value. This concept is known as **Alternative Quote Operator**(q).

We can use **any character** such as **{**, **<**, **(**, **[**, **!** or any character as **delimiter**. These characters are known as **delimiters**.

1 another example

:

Without using Quote Operator:

```
Here we get Error since we are using apostrophe in our literal value directly.

Error code below:
SELECT id, name, dept, name||' work's in '||dept||'
 department' AS "work" FROM myTable2
```

Using Quote Operator:
SELECT id, name, dept, name||q'[work's in ']'||dept||'
department' AS "work" FROM myTable2

```
Output:
See, we are able to use apostrophe in the
work column as a literal value of myTable2
```

```
ID NAME      DEPT           work
1  RR        Executive  RR work's in 'Executive department
2  GFG       HR         GFG work's in 'HR department
3  Steve     Sales      Steve work's in 'Sales department
4  Bhuvi     CSE        Bhuvi work's in 'CSE department
```

Here above see, **q'[** indicates starting of our literal value and the we use **]'** which indicate end of our literal value. So see here we have used **apostrophe** in our literal value easily(means we easily use **'s** in work) without any error that's why we get output as RR **work's** in Executive Department.]

Here above we use **[** as delimiter so it is **not** a limitation in using delimiter means we can use any character as delimiter.

References:

About Alternative Quote Operator,

Performing SQL Queries Online

Last Updated : 21 Mar, 2018

6

Similar Reads

1. What is Alternative to _N_ in PROC SQL?
2. QUOTE () function in MySQL
3. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
4. Configure SQL Jobs in SQL Server using T-SQL
5. SQL | BETWEEN & IN Operator
6. SQL | MINUS Operator
7. SQL | NOT Operator
8. SQL | Concatenation Operator
9. Difference between = and IN operator in SQL
10. SQL | UNION Operator

Previous Next