Beware of NULL values

August 02, 2022

Using common sense and critical thinking is crucial in our industry, especially once you reach a senior IT position. You want things to make sense, and when troubleshooting a problem, it usually helps to stay grounded in comparing how things are with how things should be. However, in some rare cases, you can come across things that don't quite make sense in the real world, even though they are normal in the tech sphere. This is an example of such a case, and why you need to be careful when using SQL without understanding all the special use cases that come with it.

In our scenario, we had a database table that an application was querying. The table had a column called "sold" containing an integer value. That value could be 0 or 1, although technically since it's an integer, it could also contain any other numeric value as well. The application checked whether the value was equal to 1, or whether it was not. Here are the SQL queries in question along with the results:

```
MariaDB [db]> select count(*) from inventory where sold = 1;
+----------+
| count(*) |
+----------+
|       23 |
+----------+
1 row in set (0.00 sec)

MariaDB [db]> select count(*) from inventory where sold != 1;
+----------+
| count(*) |
+----------+
|       97 |
+----------+
1 row in set (0.00 sec)
```

Now these seem like very simple queries that should always work, however it quickly became obvious that there was an issue. When adding both use cases, the total amount did not match the expected value. This can be confirmed when doing the following SQL query:

```
MariaDB [db]> select count(*) from inventory;
+----------+
| count(*) |
+----------+
|      418 |
+----------+
1 row in set (0.00 sec)
```

As you can see, the amount of rows where 'sold' is 1, added to the number of rows where 'sold' is not 1, does not equate to the total number of rows in the table. From a common sense point of view, this makes no sense at all. It's like saying you place 10 coins on a table, count the number of heads, add the number of tails, and it doesn't equal 10. But the world of SQL has a special use case that must be taken into account. That use case is the NULL value. In SQL, NULL equates no conditional query. That means any value set to NULL will not equal 1, but it also won't equal not-1. It has to be processed separately. Sure enough, if you query for NULL values and add up the number, then you get the expected count of 418:

```
MariaDB [db]> select count(*) from inventory where sold is null;
+----------+
| count(*) |
+----------+
|      298 |
+----------+
1 row in set (0.00 sec)
```

The root cause analysis showed that when the column was added to the table, all values were set to NULL by default. The solution was to change all NULL values to 0's, and then the values seen in the application started to make sense. This is a typical case of common sense leading you astray, because the mathematical truth that "x = 1" and "x != 1" should encompass all possible values of x just doesn't hold true in the world of databases.