Instead of trying to create a filter that includes everything, try just omitting the filter

devblogs.microsoft.com/oldnewthing/20121107-00

November 7, 2012

Raymond Chen

The question was sent to a peer-to-peer discussion group for an internal program, let's call it Program Q. I'll add additional context so you can follow along.

> Hi, I'm trying to build a query that finds all issues owned by a particular user, regardless of which product the issue belongs to. I know that I can query for specific products by saying
>
> ```
> q select -owner bob -product LitWare
> q select -owner bob -product Contoso
> ```
>
> Is there a better way to do this than just running the query for every product in the database? It would be great to find all the issues at one shot instead of having to issue dozens of commands and combine the results.

The person who submitted this question got so distracted by the `-product` filter, that they forgot that they could just omit the filter.

```
q select -owner bob
```

If you don't filter by product, then it finds everything regardless of the product.

Enumerating all the products, then repeating the query for each product is some sort of anti-pattern. I don't know if it has a name, so I'll make one up. The *long division* anti-pattern performs an operation on a collection by arbitrarily breaking the collection into groups, then performing the operation on each member of each group, all this even though the grouping is unrelated to the operation.

In C#, it would be phrased something like this:

```
public void MakeUnavailable(string productId)
{
    var stores = Inventory.Select(p => p.Store).Distinct();
    foreach (var store in stores) {
        foreach (var product in
                 from p in Inventory
                 where p.Store == store &&
                       p.ProductId == productId) {
            product.Available = false;
        }
    }
}
```

In words, we first dig through the inventory and collect all the unique stores. For each store, we go through the inventory again, looking for products from that store, and if the product is the one we want to make unavailable, set the `Available` property to `false`. (To avoid playing favorites, I used both fluent and query expression syntax.)

Assuming the order in which the product is made unavailable is not important (and it doesn't appear to be, since we didn't sort the stores), the grouping by store is superfluous. You can just iterate across the entire inventory without regard for store:

```
public void MakeUnavailable(string productId)
{
    foreach (var product in
             from p in Inventory
             where p.ProductId == productId
             select p) {
        product.Available = false;
    }
}
```

Raymond Chen

Follow