Force.com Formula Fields, Indexes, and Performance Gotchas

Understanding Force.com internals can give you an edge in building applications
that perform well, especially when your organization has a large volume of data.
This blog post can help you get that edge. Read on to better understand formula fields, field indexes, and date field logic so that you can deliver SOQL queries, list views, and reports that fly rather than crawl.

Understanding Force.com internals can give you an edge in building applications that perform well, especially when your organization has a large volume of data. This blog post can help you get that edge. Read on to better understand formula fields, field indexes, and date field logic so that you can deliver SOQL queries, list views, and reports that fly rather than crawl.

Scenario: Force.com Formula Fields That Hide Complexity

In many customer implementations and custom Force.com applications, I often see objects with a formula field that developers and report builders have used to simplify otherwise complex filter conditions for SOQL queries, list views, and reports. This approach is especially common when dynamic, relative date logic is used in a filter condition.

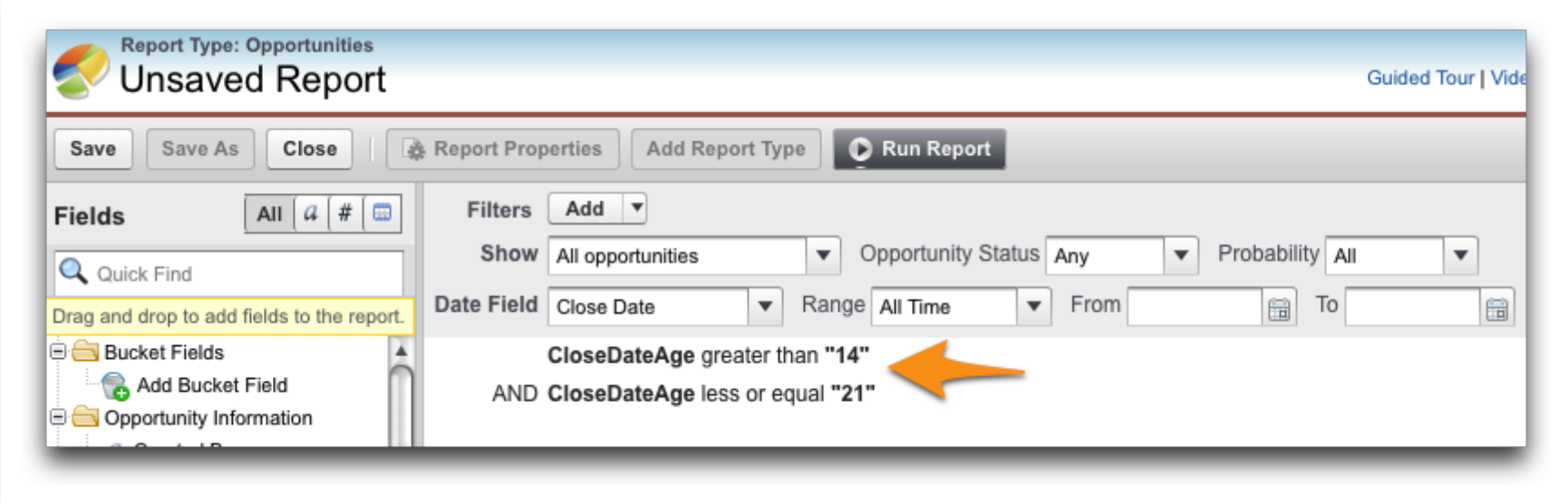
As an example, consider a custom formula field in the Opportunity object, `CloseDateAge`, with the following formula.

```
TODAY() - CloseDate
```

With the `CloseDateAge` field in place, you can write a fairly simple SOQL query filter condition to get the Opportunity records from within a specific date range, such as from 14 to 21 days ago.

```
SELECT Id, Name
  FROM Opportunity
 WHERE CloseDateAge__c >  14
   AND CloseDateAge__c <= 21
```

When you build list views and reports, you can also specify an intuitive filter condition using the formula field.



Everyone is happy because they can get the data they need with very little effort, right? Not so fast.

Consideration: Force.com Formula Fields and Indexes

By default, Force.com formula fields don't have indexes. So when you create and use a formula field such as `CloseDateAge` with very large objects (say, objects that have more than one million records), the SOQL queries and reports using the formula field as a filter might perform slower because your queries and reports have to perform full scans to find target records.

Possible Solution for Existing Formula Fields: A Custom Index

When you already have a formula field with many dependent SOQL queries and reports, consider using a custom index for the field. As a recent Technical Enablement blog post mentions, you can request salesforce.com Customer Support to create a custom index on a deterministic formula field.

Before requesting an index for a formula field, carefully consider the tradeoffs. Every index you create requires maintenance overhead when DML operations update field values that are referenced by the underlying formula. This overhead can slow the performance of bulk data loads that use the formula field in some way.

Solution for Non-Deterministic Formula Fields: Logic in Filters

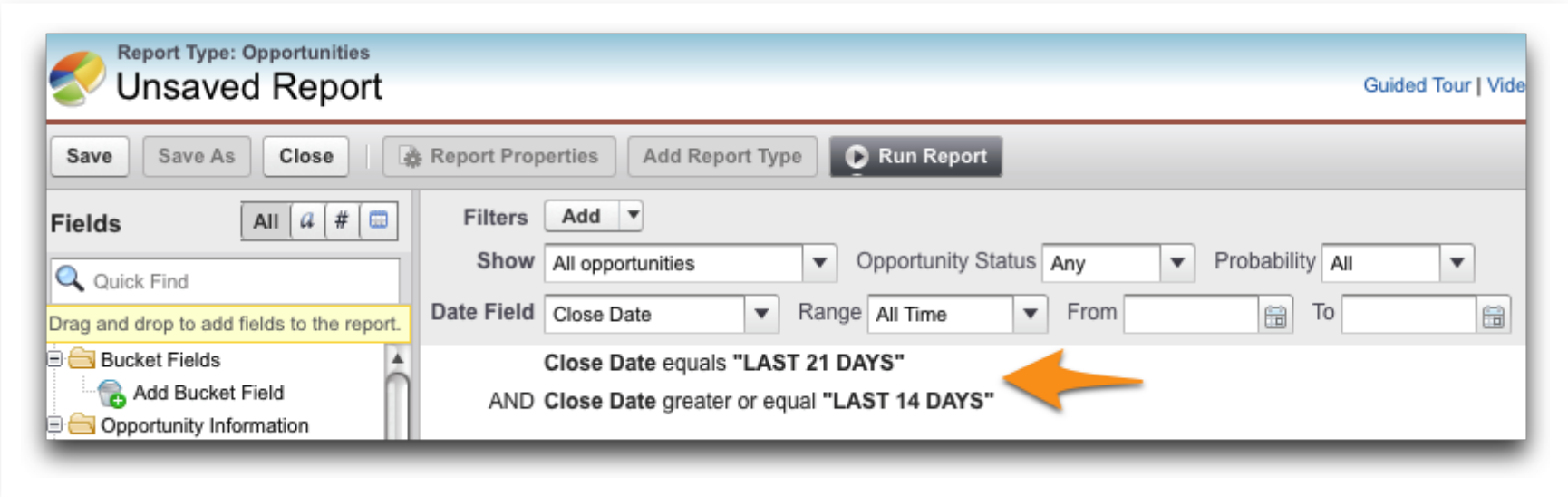
The `CloseDateAge` field is a perfect example of a non-deterministic formula field that Force.com cannot index. The field is not deterministic because the output value depends on the value of the current day, which changes every 24 hours, and factors in time zone differences. If you need a better understanding of what makes fields deterministic and non-deterministic–and how those field types affect indexing–read Force.com SOQL Best Practices: Nulls and Formula Fields.

So what can you do when you can't create an indexed formula field to hide complex filter logic? Go back to square one: Put your filter logic in your filters! There's an extra bonus with this approach–Force.com's query optimizer can consider using indexes that are already in place for fields that are in the filter logic.

For example, when filters use the `CloseDateAge` formula field, the Force.com query optimizer doesn't consider the index on the underlying Opportunity.CloseDate field. But when you use a SOQL query such as the following one, which is equivalent to the previous query, the Force.com query optimizer can consider taking full advantage of that index.

```
SELECT Id, Name
  FROM Opportunity
 WHERE ((CloseDate = LAST_N_DAYS:21)
   AND  (CloseDate < LAST_N_DAYS:14))
```

Likewise, here's an equivalent report filter condition that uses the underlying index.



Notice here that the report's filter conditions use relative date functions and are similar to the conditions from the previous SOQL query example. You can also enter such filter conditions when building list views.

Closing Thoughts

Before you create a formula field to hide complex filter condition logic, ask yourself the following questions.

- Is the encompassing object large (i.e., containing more than one million records)?

- Is the proposed formula field non-deterministic?

If the answer to both questions is “yes,” then your SOQL queries, list views, and reports might be able to perform better when they directly embed the filter logic in their definition, not when they rely on a formula field that simplifies the logic but cannot be indexed.