Tips for Reducing Formula Size

Salesforce, Winter '16

@salesforcedocs

Last updated: December 10, 2015

CONTENTS

TIPS FOR REDUCING FORMULA SIZE

In Salesforce, formulas are used in many types of business logic, including formula fields, default values for fields, workflow rules, and validation rules. Use formulas to calculate something, combine data from multiple fields, check for a condition, or show information from one record on a related record. Formulas can be simple or complex, ranging from basic strings or numbers to complex structures with layers of nested logic extracting data from multiple fields.

Formula size is made up of two parts. The first is the formula syntax that you write in the formula editor and save as a formula field, validation rule, and so on. The second is the database query that's compiled from your formula syntax and executed using your data at runtime. The query can be much larger than the formula syntax that generates it, because it requests data from all the fields involved, performs operations, and in many cases performs error checking on resulting values. Think of a formula like an iceberg: the visible part is your formula syntax, but there is a big piece below the surface that you can't see.

Formula size problems come in two flavors: the formula is too long, or its compiled size is too big. Let's look at some tips for preventing these issues.

REDUCING THE LENGTH OF YOUR FORMULA

Formula syntax is limited by the maximum size of the database field that stores it. For any Salesforce edition, you'll get an error message if you exceed these limits:

- Maximum number of characters: 3,900 characters
- Maximum formula size when saved: 4,000 bytes

When your formula exceeds one of these limits, the error message either says, "Formula is too long (3,902 characters). Maximum length is 3,900 characters." or "The size of your string is 5,385 bytes but may not exceed 4,000 bytes." This means the text that you typed into the formula editor has more characters than the database can store in one formula field. Character count includes spaces, carriage returns, and comments. If your formula includes multi-byte characters, such as in Japanese or other multi-byte languages, the number of bytes is different than the number of characters.

To correct formula limit errors, move parts of the formula into one or more secondary formula fields, and reference those in your main formula. The secondary fields don't need to appear on the page layout.

To figure out which bits of your formula to break out, look for large blocks of text or expressions that are repeated multiple times. Breaking up a formula can also make it easier to understand if the secondary formula fields are named well.

For example, suppose you have a Membership Start Date field, and you're using that date to calculate a Membership End Date (in the interest of space, we'll keep it short, but pretend it exceeds 3,900 characters):

```
IF(
  MONTH( Membership_Start_Date__c ) < 10,
  DATE( YEAR( Membership_Start_Date__c ), 12, 31),
  DATE( YEAR( Membership_Start_Date__c ) + 1, 12, 31)
)
```

Because you repeatedly reference `YEAR(Membership_Start_Date__c)`, you can move that into a formula field called Start Year. You can also move `MONTH(Membership_Start_Date__c)` into its own formula field.

Then the main formula becomes:

```
IF(
  Start_Month__c < 10,
  DATE( Start_Year__c, 12, 31),
  DATE( Start_Year__c + 1, 12, 31)
)
```

You're now down to 89 characters from 146. Since there are edition-based limits on how many fields you can create for an object, take them into account when figuring out how to break up a long formula. You might only want to create `Start_Year__c` in the example above.

Keeping comments and field names as short as possible while still being descriptive can also help. You might want Membership Start Date to be the label for your users, but you could conserve space by overriding the default API name "Membership_Start_Date" to "Start_Date" or "Mem_Start_Date". Override the default name when you create a custom field, rather than trying to change it later.

You can remove blank spaces and carriage returns, but this makes complex formulas harder to read, so consider that a last resort.

REDUCING YOUR FORMULA'S COMPILE SIZE

The query that's compiled from your formula syntax is limited by the maximum query size that the database can execute. This limit is the same for all Salesforce editions: Maximum formula size (in bytes) when compiled: 5,000 bytes.

When your formula exceeds this limit, the error message says, "Compiled formula is too big to execute (13,974 characters). Maximum size is 5,000 characters." This means that the query generated by your formula syntax is too big for the database to handle all at once.

Reducing the number of characters in your formula doesn't help: comments, white space and field name length make no difference on the compile size. Breaking the formula into multiple fields doesn't help either because the compile size of each field is included in the main formula's compile size.

Fortunately there are ways to work around this limit, and you can avoid it in many cases by making your formulas more efficient.

Minimize the number of references to other fields

The most important thing you can do to reduce your formula compile size is reduce the references to other formula fields. Each time you reference a field, the compile size of that field is added to your current formula. A simple field type like a Date is small, but for other formula fields, the size can add up.

Consider two fields:

- `Date1__c` is a Date field.
- `Date2__c` is a formula field that creates a new date from `Date1__c`:

```
DATE( YEAR( Date1__c ), MONTH( Date1__c ), DAY( Date1__c ) )
```

Using `Date1__c` in a formula adds 22 characters to the compile size each time it's referenced. `Date2__c` adds 465 characters each time. Let's look at the impact on this formula that generates a deadline two business days after a given date:

```
CASE( MOD( SomeDate__c - DATE( 1900, 1, 7 ), 7 ),
 0, SomeDate__c + 1 + 2, /* Sunday */
 1, SomeDate__c + 2,     /* Monday */
 2, SomeDate__c + 2,     /* Tuesday */
 3, SomeDate__c + 2,     /* Wednesday */
 4, SomeDate__c + 2 + 2, /* Thursday */
 5, SomeDate__c + 2 + 2, /* Friday */
 6, SomeDate__c + 2 + 2, /* Saturday */
 SomeDate__c             /* Default */
)
```

Replacing `SomeDate__c` in the above formula with `Date1__c` gives us a formula compile size of 487 characters. But replacing `SomeDate__c` with `Date2__c` compiles to 4,474 characters! Most of that is the nine references to `Date2__c`: 9 * 465 = 4,185 characters.

So how can we reduce the number of times we reference other fields?

Leverage the "default" value in `CASE()`

The last argument in the `CASE()` function is the default value. If you have a lot of cases with the same value, use it as the default to reduce the number of checks. Let's take another look at the deadline formula:

```
CASE( MOD( SomeDate__c - DATE( 1900, 1, 7 ), 7 ),
 0, SomeDate__c + 1 + 2, /* Sunday */
 1, SomeDate__c + 2,     /* Monday */
 2, SomeDate__c + 2,     /* Tuesday */
 3, SomeDate__c + 2,     /* Wednesday */
 4, SomeDate__c + 2 + 2, /* Thursday */
 5, SomeDate__c + 2 + 2, /* Friday */
 6, SomeDate__c + 2 + 2, /* Saturday */
 SomeDate__c             /* Default */
)
```

The default value in this formula is `SomeDate__c`. But the value of `MOD( Date__c - DATE( 1900, 1, 7 ), 7 )` is always 0 to 6, so the default value is never used. This formula could be rewritten as:

```
CASE( MOD( SomeDate__c - DATE( 1900, 1, 7 ), 7 ),
 0, SomeDate__c + 1 + 2, /* Sunday */
 4, SomeDate__c + 2 + 2, /* Thursday */
 5, SomeDate__c + 2 + 2, /* Friday */
 6, SomeDate__c + 2 + 2, /* Saturday */
 SomeDate__c + 2         /* Default - Mon/Tues/Wed */
)
```

By making the Monday/Tuesday/Wednesday case the default (date + 2 days), we reduce the compile size to 360 characters for `Date1__c`, and to 3,018 for `Date2__c`.

Use `CASE()` instead of nested `OR()` statements

The following formula returns the date of the last day of the month for a given date (assume February always has 28 days):

```
DATE(
 YEAR( SomeDate__c ),
 MONTH( SomeDate__c ),
 IF(
   OR(
     MONTH( SomeDate__c ) = 4,
     MONTH( SomeDate__c ) = 6,
     MONTH( SomeDate__c ) = 9,
     MONTH( SomeDate__c ) = 11
   ),
   30,
   IF(
     MONTH( SomeDate__c ) = 2,
     28,
     31
   )
 )
)
```

The formula first checks for months with 30 days, then February, and the remaining months are 31 days. It requires a nested `IF()` function, which isn't very readable and compiles to 1069 characters for `Date1__c` and a whopping 7,271 characters for `Date2__c`! Why? Because the formula references the date seven times. Compare that to this revised version of the formula:

```
DATE(
 YEAR( SomeDate__c ),
 MONTH( SomeDate__c ),
 CASE(
   MONTH( SomeDate__c ),
   2, 28,
   4, 30,
   6, 30,
   9, 30,
   11, 30,
   31
 )
)
```

Not only is this easier to read, the formula compiles to only 645 characters for `Date1__c` and 3,309 characters for `Date2__c`, and it now references the date three times instead of seven.

Rearrange the logic

This example came from the Salesforce Answers community. A picklist stores the name of an agent responsible for an opportunity. The formula calculates a commission, based on Base Commission value and a multiplier. But because `Base_Commission__c` is referenced in each condition of the `CASE()` statement, the formula exceeds the compile size.

```
CASE( Agent__c,
  "John", Base_Commission__c * 2,
  "Jane", Base_Commission__c * 6,
  /* Repeat for many other agents */
  Base_Commission__c
)
```

To fix this, move `Base_Commission__c` outside the `CASE()` function. The formula can be rewritten as:

```
Base_Commission__c * CASE( Agent__c,
  "John", 2,
  "Jane", 6,
  /* Repeat for many other agents */
  1
)
```

Even if Base Commission is only a Currency field and not a formula itself, referencing it once instead of multiple times greatly reduces the formula compile size.

As another example, let's try this with our business days formula:

```
SomeDate__c + CASE( MOD( SomeDate__c - DATE( 1900, 1, 7 ), 7 ),
  0, 1 + 2, /* Sunday */
  4, 2 + 2, /* Thursday */
  5, 2 + 2, /* Friday */
  6, 2 + 2, /* Saturday */
  2         /* Default - Mon/Tues/Wed */
)
```

We have now further reduced the size to 188 characters for `Date1__c` and to 1,074 characters for `Date2__c`, which is almost a quarter of the original formula size.

Minimize the number of times formula functions are called

All formula functions are not compiled equally. Date-related functions in particular—like `DATE()` and `MONTH()`—generate large queries. The fewer times you need to reference functions, the smaller your compile size is likely to be.

We just saw an example where we rearranged the logic in a `CASE()` statement to reduce the number of times a field was referenced. This strategy also works for formula functions.

Here's another example from the Answers community. A picklist contained a list of airlines. The formula returned a link to the airline's website and looked something like this:

```
CASE( Airline__c,
  "Airline 1",  HYPERLINK("http://airline1.com", "Airline 1"),
  "Airline 2",  HYPERLINK("http://airline2.com", "Airline 2"),
  /* Dozens of other airlines */
  ""
)
```

There are so many options in the picklist, the formula hits the compile size limit. But since the friendly name for the link is the same as the picklist text, moving `HYPERLINK()` outside the `CASE()` statement allows it to be referenced just once:

```
HYPERLINK(
  CASE( Airline__c,
    "Airline 1", "http://airline1.com",
    "Airline 2", "http://airline2.com",
    /* Dozens of other airlines */
    ""
  ), /* CASE() generates the URL for the hyperlink */
  TEXT(Airline__c) /* TEXT() generates the friendly name */
)
```

The revised formula adds only 38 characters to the compile size for each picklist option, instead of 133.

Rethink your picklist

In the airline and agent commission examples above, `CASE()` is used to check each value in a picklist to set a formula field value. If you have more than 20 picklist options and you plan to check each possible condition in a `CASE()` statement to determine another value, consider a lookup instead of a picklist.

When choosing between picklist and lookup, think about:

- How many objects your Salesforce edition supports
- Whether you have dependencies on other field values
- The experience for the end user
- How frequently you change the picklist values
- How close you are to hitting the cross-object reference limit

If you can't reduce the compile size of this type of formula using other methods, consider making the picklist a lookup to a custom object with the name as the picklist value, then create a custom field on that object for the value you need to set in the formula.

In the commission example we looked at earlier, if the `Agent` field is a lookup to an Agent object with a numeric field called `Commission Multiplier`, the formula becomes a simple calculation using a cross-object reference:

```
Base_Commission__c * Agent__r.Commission_Multiplier__c
```

Added bonus: when you add new agents, you don't have to remember to change the formula. It just works.

Think about the problem another way

A common formula to format the elapsed time between two Date/Time fields to days, hours, and minutes looks like this (where `Diff__c` is one Date/Time subtracted from another to get the difference in days):

```
IF (
  Diff__c > 0,
  TEXT(FLOOR(Diff__c)) & " days "
  & TEXT(FLOOR(24 * (Diff__c - FLOOR(Diff__c)))) & " hours "
  & TEXT(ROUND(60 * (ROUND(24 * (Diff__c - FLOOR(Diff__c)), 8)
    - FLOOR(ROUND(24 * (Diff__c - FLOOR(Diff__c)), 8))), 0))
    & " minutes ",
  ""
)
```

This formula checks if the difference is a positive value, and if it is, several operations calculate values for days, hours, and minutes in `Diff__c` and return a string such as "6 days 2 hours 11 minutes." This formula compiles to 2,547 characters. And it works fine when `Diff__c` subtracts two simple Date/Time fields like this:

```
LastModifiedDate - CreatedDate
```

However, if `Diff__c` is a more complex formula, the elapsed time formula becomes too big to compile. Why? Count the number of times `Diff__c` is referenced. Also, all those nested `FLOOR()` and `ROUND()` functions add up.

There's a simpler way to think about the problem:

```
Number of days = DateTime1 - DateTime2
Number of hours = Number of days * 24
Number of minutes = Number of hours * 60
```

The `MOD()` function can really help. The modulus of the number of hours divided by 24 is the number of hours not accounted for by days. The modulus of number of minutes divided by 60 is the number of minutes not accounted for by hours. So we can change the formula to:

```
IF(
  Diff__c > 0 ,
  TEXT(FLOOR(Diff__c)) & " days "
    & TEXT(FLOOR(MOD(Diff__c * 24, 24))) & " hours "
    & TEXT(ROUND(MOD(Diff__c * 24 * 60, 60), 0)) &" minutes",
  ""
)
```

The new version of the formula compiles to 728 characters (down from 2,547), and because it only includes `Diff__c` four times, it has room to accommodate more complex formulas in that field.

Here's another example. This validation rule formula returns true if an opportunity's Close Date is not in the current month

```
OR (
CloseDate < DATE( YEAR(TODAY()), MONTH(TODAY()), 1),
IF (
AND (
MONTH (TODAY() ) =1,
CloseDate > DATE( YEAR(TODAY() ), MONTH(TODAY() ), 31)),
true,
```

```
IF (
AND (
MONTH (TODAY() ) =2,
CloseDate > DATE( YEAR(TODAY() ), MONTH(TODAY() ), 28)),
true,
IF (
AND (
MONTH (TODAY() ) =3,
CloseDate > DATE( YEAR(TODAY() ), MONTH(TODAY() ), 31)),
true,
IF (
AND (
MONTH (TODAY() ) =4,
CloseDate > DATE( YEAR(TODAY() ), MONTH(TODAY() ), 30)),
true,
IF (
AND (
MONTH (TODAY() ) =5,
CloseDate > DATE( YEAR(TODAY() ), MONTH(TODAY() ), 31)),
true,
IF (
AND (
MONTH (TODAY() ) =6,
CloseDate > DATE( YEAR(TODAY() ), MONTH(TODAY() ), 30)),
true,
IF (
AND (
MONTH (TODAY() ) =7,
CloseDate > DATE( YEAR(TODAY() ), MONTH(TODAY() ), 31)),
true,
IF (
AND (
MONTH (TODAY() ) =8,
CloseDate > DATE( YEAR(TODAY() ), MONTH(TODAY() ), 31)),
true,
IF (
AND (
MONTH (TODAY() ) =9,
CloseDate > DATE( YEAR(TODAY() ), MONTH(TODAY() ), 30)),
true,
IF (
AND (
MONTH (TODAY() ) =10,
CloseDate > DATE( YEAR(TODAY() ), MONTH(TODAY() ), 31)),
true,
IF (
AND (
MONTH (TODAY() ) =11,
CloseDate > DATE( YEAR(TODAY() ), MONTH(TODAY() ), 30)),
true,
IF (
AND (
MONTH (TODAY() ) =12,
CloseDate > DATE( YEAR(TODAY() ), MONTH(TODAY() ), 31)),
```

```
true, false
))))))))))))
```

This complex check determines whether the Close Date is earlier than the first day of the current month, or later than the last day of this month. But actually, the day doesn't matter at all: if the month and year of the Close Date are the same as the month and year of the current month, then it's the same month. So we can rewrite the formula as:

```
NOT(
  AND(
    MONTH( CloseDate ) = MONTH( TODAY() ),
    YEAR( CloseDate ) = YEAR( TODAY() )
  )
)
```

This new version is much more readable, and only compiles to 200 characters compared to more than 3,000 for the original formula.

If all else fails, use a workflow field update

Sometimes there's just no way to rework a formula to make it compile small enough. For example, on the Answers community, someone asked for help with a formula that compiled to more than 45,000 characters. Reducing references to other formula fields and reducing the size of those referenced formulas only got the size down to about 32,000 characters.

If you have Enterprise Edition, Unlimited, or Performance Edition, you can use a workflow rule with a field update action to set the value of a regular custom field using a formula. When you reference that custom field in a formula, only the value is returned—not the formula that created it. Here's how to do it:

1. Create a custom field of the type your field update formula will return, such as Date or Number. Don't add it to any page layouts.
2. Create a workflow rule on the object. Set it to execute whenever a record is created or updated.
3. Create a Field Update workflow action. Choose the custom field you created in Step 1 as the target, and copy part of your large formula into the formula that sets the value in that field.
4. In your formula field, replace the part of the formula you copied to the field update action with a reference to your custom field.

Remember the formula that returned the last day of the month for a given date, and how when the formula field `Date2__c` was substituted for `SomeDate__c` the formula was over the compile size limit?

```
DATE(
 YEAR( SomeDate__c ),
 MONTH( SomeDate__c ),
 IF(
   OR(
     MONTH( SomeDate__c ) = 4,
     MONTH( SomeDate__c ) = 6,
     MONTH( SomeDate__c ) = 9,
     MONTH( SomeDate__c ) = 11
   ),
   30,
   IF(
     MONTH( SomeDate__c ) = 2,
     28,
     31
   )
 )
)
```

Another way we could have solved this is by making `Date2__c` a custom Date field instead of a formula field, and creating a workflow rule and field update to set `Date2__c` to the value of the formula it previously contained

```
DATE( YEAR( Date1__c ), MONTH( Date1__c ), DAY( Date1__c ) )
```

Now the size of `Date2__c` is the same as `Date1__c`, because it contains a simple date, and the compile size of the last-day-of-the-month formula is 1069 characters instead of over 7,000.

NEED MORE HELP?

We hope these examples help when you hit one of the formula size limits. If you get stuck, try posting your question to the very smart people on the Salesforce Answers community, the Formulas & Validation Rules Discussion on the Salesforce Developers discussion boards, or the Salesforce Stack Exchange. Chances are someone else has run into the same thing, and often it just takes another pair of eyes looking at a problem to solve it.