This topic describes two ways in which you can improve macro performance in scoring and contact groups:

1. Create a custom macro rule translator - by default, custom macro rules that you use in scoring and contact group conditions evaluate individually for each contact. You can improve the recalculation performance by implementing and registering translators for these macro rules. Once a macro rule has a translator registered, a database query is constructed that then selects only the contacts that fit the given macro condition.
2. Limit unnecessary recalculations - you can further improve your site performance by making sure your custom contact group and scoring macro rules are recalculated only when users perform activity types that are relevant to the macro rule.

Default macro rules

Macro rules available in Kentico by default have both these optimizations implemented out of the box.

Creating a custom macro rule translator

1. Create a class that implements the *IMacroRuleInstanceTranslator* interface.
2. Implement the logic of the macro rule in a method that returns a query of the *ObjectQuery<ContactInfo>* type. The returned object needs to contain the contacts that fulfill the condition. Note that you do not have to handle contextual information, such as site context, in the query.

Example of an implemented ContactIsFemaleMacro rule translator

```
using System;

using CMS.DataEngine;
using CMS.Membership;
using CMS.ContactManagement;

internal class ContactIsFemaleInstanceTranslator: IMacroRuleInstanceTranslator
{
    /// <summary>
    /// Translates ContactIsFemaleMacro rule.
    /// </summary>
    public ObjectQuery<ContactInfo> Translate(MacroRuleInstance macroRuleInstance)
    {
        if (macroRuleInstance == null)
        {
            throw new ArgumentNullException("macroRuleInstance");
        }
        if (macroRuleInstance.MacroRuleName != "ContactIsFemaleMacro")
        {
            throw new ArgumentException("[ContactIsFemaleTranslator.Translate]:
Only macro rule instances of type ContactIsFemaleMacro can be translated");
        }

                var ruleParameters = macroRuleInstance.Parameters;
                // ruleParameters contains the data user enters in macro fields
        string paramIs = ruleParameters["_is"].Value;
        QueryOperator queryOperator = paramIs == "!" ? QueryOperator.NotEquals :
QueryOperator.Equals;

        return ContactInfoProvider.GetContacts().Where("ContactGender",
queryOperator, (int)UserGenderEnum.Female);
    }
}
```

3. Register an instance of the *MacroRuleMetadata* class by calling the *MacroRuleMetadataContainer.RegisterMetadata* method.

   - To register the class at the beginning of the application's lifecycle, create a custom module class and override the **OnInit** method.
   - Either add the module class into a custom project within the Kentico solution (recommended) or directly into the Kentico web project (into a custom folder under the **CMSApp** project for *web application* installations, into the **App_Code** folder for *web site* installations).

```
using System.Collections.Generic;

using CMS;
using CMS.Base;
using CMS.DataEngine;
using CMS.ContactManagement;

// Registers the custom module into the system
[assembly: RegisterModule(typeof(CustomInitializationModule))]

public class CustomInitializationModule : Module
{
        // Module class constructor, the system registers the module under
the name "CustomInit"
        public CustomInitializationModule()
                : base("CustomInit")
        {
        }

        // Contains initialization code that is executed when the
application starts
        protected override void OnInit()
        {
                base.OnInit();

                MacroRuleMetadata metadata = new MacroRuleMetadata(
                        "ContactIsFemaleMacro",
                        new ContactIsFemaleInstanceTranslator(),
                        affectingActivities: new List<string>(0),
                        affectingAttributes: new List<string>(0));

                MacroRuleMetadataContainer.RegisterMetadata(metadata);
        }
}
```

4. Save the files.
   Important: Note that this way, the recalculation is not performed for any contact activity or attribute change. You can make the rule recalculate for a specific set of activities and attribute changes.

Now, when you use the translated macro rule in a scoring or contact group condition, the query is only performed once when you run recalculation.

> **Compound conditions**
>
> All macro rules used in a contact group or scoring macro conditions need to be translated. Otherwise, the evaluation will be performed in a non-optimized way.

Recalculating contact group and scoring rules only for specific activities and attributes

Another way of improving your site performance is by making sure that contact groups and scores are recalculated only when necessary. You can make sure your custom contact group and scoring macro rules are only recalculated for specified activities or when specific contact attributes change.

Examples:

- The "ContactIsFemale" macro doesn't need to be rebuilt by any contact activity that the visitor performs on a site. Instead, it only needs to be rebuilt by a change in the contact's "contactgender" attribute.
- A rule that checks whether a contact is registered for an event ("CMSContactIsRegisteredForSpecifiedEvent" macro by default) only needs to be rebuilt when a visitor performs an event booking activity on a site ("EVENT_BOOKING" activity by default).
- A rule that checks whether a contact has visited a page ("CMSContactHasVisitedSpecifiedPageInLastXDays" macro by default) only needs to be rebuilt when a visitor performs an event booking activity on a site ("PAGE_VISIT" activity by default).

To specify the activities and attribute changes that rebuild your custom macro rule:

1. Create a custom macro rule translator.
2. Extend the metadata instantiation by the following parameters:
    a. **IList<string> affectingActivities** - a list of activities for which you want the macro to rebuild.
    b. **IList<string> affectingAttributes** - a list of attributes for which you want the macro to rebuild.

Rebuild custom macro rule for specific activities or attribute changes

```
protected override void OnInit()
{
        ...
        MacroRuleMetadata metadata = new MacroRuleMetadata(
                "CustomMacroRule",
                new CustomMacroRuleTranslator(),
                affectingActivities: new List<string>
                {
                        // Recalculate for a predefined activity
                        PredefinedActivityType.PAGE_VISIT,
                        // Recalculate for a custom activity
                        "custom_activity_codename"
                }
                affectingAttributes: new List<string>
                {
                        // Recalculate when specific attributes change
                        "contactgender",
                        "contactcountryid"
                });
        MacroRuleMetadataContainer.RegisterMetadata(metadata);
        ...
}
```

Do not rebuild custom macro rule for any activities or attribute changes

```
protected override void OnInit()
{
        ...
        var noNeedToRecalculate = new List<string>(0);

        MacroRuleMetadata metadata = new MacroRuleMetadata(
                "CustomMacroRule",
                new CustomMacroRuleTranslator(),
                affectingActivities : noNeedToRecalculate,
                affectingAttributes : noNeedToRecalculate
                );
        MacroRuleMetadataContainer.RegisterMetadata(metadata);
        ...
}
```

Rebuild custom macro rule for all activities and attribute changes

```
protected override void OnInit()
{
        ...
        MacroRuleMetadata metadata = new MacroRuleMetadata(
        "CustomMacroRule",
        new CustomMacroRuleTranslator(),
        affectingActivities: new List<string>
                {
            MacroRuleMetadata.ALL_ACTIVITIES
                },
                affectingAttributes: new List<string>
                {
                        MacroRuleMetadata.ALL_ATTRIBUTES
                });
    MacroRuleMetadataContainer.RegisterMetadata(metadata);
        ...
}
```

Note: You can also combine the parameters as you see fit.

3. Save the files.

Now, only the specified contact activities or changes in contact attributes will trigger a recalculation of the custom macro rule.