

# Optimising Performance in Siebel

By Duncan Scattergood, Operations Director, Customer Systems plc

Siebel is an extremely powerful application that has revolutionised the way companies build relationships with their customers. With the right implementation skills and expertise, Siebel is a dependable application with excellent performance. However, mistakes in design and/or configuration can lead to particular functions, parts of or even all of the application running slower than might be expected.

Often these performance problems can be swiftly remedied if the right level of determination, skill and intellectual effort is applied. It is important to realise that, although performance problems can be characterised by certain common features, in fact every performance problem is slightly different, and so generalised solutions should not become a substitute for expertise.

This article will examine some common types of performance problems which are encountered in Siebel. For each of these types, we will suggest the ways in which the problems should be investigated and the approaches that can be adopted when designing solutions.

## What is unacceptable performance?

In order to consider how to eliminate it, we must first identify what constitutes unacceptable performance. This varies from application to application, and depends very much on the context in which the poor performance occurs. For example, if only one particular task is affected by poor performance, and that task is used by one user once a month, then although occasionally problematic for one user, this may not constitute unacceptable performance. However, if the task is used on a daily basis by the majority of users, then the same level of performance may be unacceptable.

Therefore, when identifying the performance issues to address in your Siebel application, it is important to consider the con-

text and impact of the slow performing elements. In particular, the following questions should be considered for each task that is affected by poor performance:

- How frequently is the task performed?
- How many users need to use the task?
- What is the impact of the slow performance of this task on the users?

## How to approach a performance improvement project

There are undoubtedly common themes among slow performing applications, and common causes of slow performance. However, it is essential not to generalise, and to recognise that every situation is different. Any project to improve performance must be approached in a systematic way, but always with an open mind about where performance problems may be found, and how they may be tackled.

The best way to identify the real problem areas within an application is to interview the users of the application, and to actually get them to show you the problem. This will enable the evaluation of the application's performance, the identification of the performance problems which are actually affecting users' abilities to do their jobs on a day-to-day basis, and an assessment of the impact each of these problems has on overall productivity. Encouraging users to actually demonstrate the problem may often lead you to find that the real issue is different from the issue that was originally reported – or that the real need is not for performance improvements but for

user/administrator training, to enable them to use the application in the optimal way.

For example, we had a recent situation where, whenever a sales person created a new opportunity, they had to add their manager manually to the sales team so that the manager could see the opportunity. If the administrator had set up the relationship between the individuals' positions correctly, there would have been no need for this manual step and the user would have been saved time and effort.

Once this investigation has been carried out, work should begin on analysing the problems that have been uncovered, and discovering their root causes. Remember that the analysis of a performance problem often takes longer than the eventual development of the solution. Some issues will require intensive intellectual effort, and others may require re-architecting of parts of the solution. Expert-level skill is indispensable at this stage. A resource with extensive experience of rooting out poorly performing configuration will pinpoint the cause far more quickly than one with little performance-specific experience.

## Major types of poor performance

As discussed previously, poor performance and its causes vary from application to application. However, there are common themes, so we will explore three major areas into which the majority of performance problems fall.

### A. The whole application runs slowly

One type of poor performance is where the entire application is found to run slowly. This particular type of performance problem is perceived to be common, but in reality it applies in probably less than 5% of cases.

When investigating a slow performing application, it is important to find out whether performance is sensitive to factors such as the number of users or the time of day. Are only particular user communities or geographic locations affected? These factors can provide enormous assistance when identifying the cause of the poor performance.

Where performance is found to be poor across the application, then causes such as the network and hardware on which the application is run should be suspected.

“The best way to identify the real problem areas within an application is to interview the users of the application, and to actually get them to show you the problem.”

Settings for the object manager and for the database should also be investigated.

However, if the nature of the application is such that most users perform only one or two common tasks, then configuration should be suspected as a possible cause of the slow performance.

Example:

We once encountered a situation where the complaint was that the whole application ran slowly for a particular group of users in a particular location. Before our involvement, everyone was focused on the fact that the users were on a different sub-LAN to the rest of the organisation and it was therefore assumed that the problem was network related. When we looked at the problem, however, we spotted that this group of users performed a single task which was not performed by anyone else in the organisation.

This then suggested to us that the problem might be the way the particular task performed and that the location of the users might be a red herring. We tested our theory by signing on as one of the users who had a problem from a location that was known to generally perform well. The task still performed badly which proved what we had suspected – that the problem related to the configuration of the task and not the network. It was then a short step to investigate the configuration, and to discover and fix the real problems.

#### B. Specific tasks or processes perform poorly

Poor performance often does not affect the entire application. In many cases, only certain interfaces perform poorly, particular views take a long time to load, or certain actions take a long time to complete. Although a limited number of tasks or processes are affected, these types of issue constitute a significant performance problem if the tasks or processes are widely or frequently used.

*Example 1: certain views take a long time to load*

If this is the type of performance problem that has been identified, then the cause of it is almost certainly one or more badly performing SQL statements. However, it is essential not to jump to conclusions as to how to deal with this scenario. Most people assume that the solution is simply to add indices to the database. It is true that this solves the problem in some cases, but in many cases it does not, or may even cause further problems. Well thought-out indices can improve query times, but there is a tradeoff because every

index represents more workload for the system when data is updated. In Siebel, it is quite common that the best solution to the problem does not involve adding more indices and, in some cases, adding more indices does not help at all.

When tackling this type of performance problem, the first thing to check is that there is only one SQL statement generated by Siebel to load the view that is affected by bad performance. Multiple SQL statements are problematic when the application is showing a list of data because there will always be a main SQL statement for the list and then one or more SQL statements for each row in the list. As there is an overhead for every SQL statement, the total overhead will grow quickly as the list grows. The configuration example below for multi value links is an example of this type of problem.

Assuming there is only one SQL statement, the SQL statement can then be simplified, for example by removing columns and tables from it systematically until the problem disappears, in order to identify the problem. Next, the configuration must be examined in order to see what part of it is generating the problematic part of the SQL statement and, finally, the configuration can be corrected so that it generates more efficient SQL.

The following sorts of configuration errors are frequently to blame for slowly loading views:

- No primaries defined on multi value links (MVLs).
- Calculations carried out on multi value fields (MVF's).
- Excessive use of ForceActive, resulting in unnecessary fields being returned in queries, potentially forcing the inclusion of extra tables in the SQL statement.
- Sorts and/or searches executed across multiple tables.

Configuration Example:  
No primaries on multi value links

A common configuration error that can lead to performance problems is not putting a primary on a multi-value link. Let's take the example of exposing Products on the Opportunity List Applet where each Opportunity can be for many Products.

If the multi value link to Product in the Opportunity Business Component is configured with no primary, then Siebel will generate a separate query over the Product Business Component for each Opportunity. In other words, if there are n Opportunities, n+ 1 pieces of SQL will be generated and executed.

By contrast, if a primary is specified, then Siebel will generate a single piece of SQL that uses the primary field to form a join to the Product table. This query is fundamentally much more efficient and will run faster, whilst not changing the way the application works from a user point of view.

*Example 2: a button click takes a long time to execute*

Usually, this type of problem occurs when a whole series of updates occur at a particular point in a process, so a script or a workflow is often involved.

The first stage of investigating this type of performance issue is to break the process down into smaller parts and then to examine it in order to track down the problem. The following are the main factors to consider during this examination:

- Is the code as efficient as possible?
- Are there any redundant queries?
- Exactly what SQL is generated?
- Are events triggering events which are then triggering events and so on? In other words, do we have the situation where an apparently efficient piece of code is calling something that is inefficient?
- Is it possible to run any parts of the workflow as an asynchronous background process, and thereby give the user a faster response even though there is still a lot of processing to be done?

*Example 3: a batch interface takes hours to execute*

Begin by breaking the interface up into stages, and then ask whether the right technology is being employed, whether the appropriate architecture (parallel or sequential) is being used, and whether there are opportunities to reduce the volumes of data, for example by spreading the load throughout the day, or by performing incremental updates instead of full data loads.

The investigation should include all of the non-Siebel parts of an interface, such as data mapping performed in a third party tool.

In the case of EIM interfaces, particular consideration must be given to batch sizes, as well as to the optimisation of the configuration (.ifb) files. For eAI interfaces, consider using simplified business components.

#### C. Particular tasks are cumbersome

Often the most dramatic improvements from a user perspective do not actually involve changing the response time of the application, but rather they involve looking at key tasks and redesigning them with the goal of minimising the number of clicks and keystrokes that a user needs to make in order to perform them. Arguably,

## “Making frequently-used tasks less cumbersome can greatly contribute to increased user satisfaction...”

these are not “performance problems”, but in our experience this is exactly the terminology used by the majority of users to describe these situations.

Making frequently-used tasks less cumbersome can greatly contribute to increased user satisfaction, and can serve to improve user perception of the application’s performance. Furthermore, the resulting configuration is often simpler and easier to maintain than the configuration it replaces.

### Summary

When embarking on a performance improvement project in Siebel, it is essential to adopt a systematic approach with an open mind, and not to jump to conclusions. However, more often than not, you should suspect poor configuration rather than inappropriate architecture. At all stages, keep in mind the real objective of the exercise: to create an application which is straightforward, responsive and fast for users. This will allow you to focus your efforts in the most important areas, and get the maximum improvement for the minimum effort.

### Case studies: succeeding where others had failed

Here we will consider two case studies which demonstrate how Customer Systems have succeeded in delivering significant improvements in performance within a matter of days.

*Customer Systems delivered considerable performance improvements for a membership organisation by redesigning the way its online registration system worked within just three days.*

This customer was about to launch its combined web and call centre application which would allow members to register and make payments, either online or by speaking to an agent.

During the test cycle, a load test had been run to ensure that the application could cope with 400 simultaneous users. However, during load testing, it was discovered that when 150 concurrent users were using the application, performance seriously deteriorated. Several user actions, such as opening views and clicking buttons, that took approximately 3 seconds to perform when only one user was using the system took 30 seconds to execute with 150 users connected.

Furthermore, when reaching 150 concurrent users, it was not possible to load any further users. The database became overloaded and the tests needed to be aborted, falling more than 50% short of the number of users required.

The customer’s efforts to resolve these performance problems had failed, and we were engaged to provide the necessary assistance.

Within a day of coming on site, our consultant had developed a working theory of what was causing the performance issues. This was suspected to be poor configuration resulting in unnecessary SQL queries being executed.

By lunchtime on the second day, our consultant had implemented a solution in the load testing environment and had run another test.

As a result of the performance improvements implemented by our consultant, the application was readily able to cope with the 400 users in the load test with no bottlenecks, compared to less than 150 without the performance improvements.

Within just three days, our consultant delivered improvements in response times such that the important views loaded 70% faster with 300 concurrent users than they had done with just a single user before our intervention.

*Customer Systems traced significant performance problems in a customer’s application to a small number of transactions, and delivered a huge improvement in the running of the application within just five days.*

This customer uses Siebel to track the testing and approval of new products.

They had experienced severe, long-standing performance issues in the application. Numerous common queries took several minutes to complete and users were forced to open additional Siebel client sessions in order to conduct work while a query executed in the original session. Furthermore, the performance was erratic and varied significantly over the working day.

The performance problems were severe enough to lead the customer to consider upgrading to a new version of Siebel in order to resolve them. They engaged us to estimate the upgrade.

After some investigation, our consultant’s conclusion was that, if an upgrade were carried out without any existing performance problems being addressed, then these issues would be migrated to the newer implementation of Siebel, and the users’ problems would not be solved.

Our suggestion was that the performance issues should be reviewed before the upgrade took place, to see whether they could be rapidly resolved for the immediate benefit of the users.

Six key transactions were identified which were commonly used, and which were exhibiting poor performance. During a five day assignment, we reduced the response time of each of these transactions from minutes down to one or two seconds. As a result of the improvement in the performance of these transactions, the “cache hit ratio” on the database (an indicator for how well tuned a database is) improved by almost a third.

Within just five days, our consultant had identified and improved these key transactions, resulting in a faster running database, and a more stable performance of the Siebel application over the course of the working day.

### About the Author



Duncan Scattergood is Operations Director and co-founder of Customer Systems plc, where he is responsible for all sales, marketing, project delivery and recruitment functions. Customer Systems is wholly focused on Siebel, Siebel Reports and Siebel Analytics and has accomplished hundreds of successful implementations. Based in the UK, they have delivered services in 28 countries across 4 continents. 19% of the customers listed in Siebel’s last annual report have used Customer Systems. Duncan can be contacted at [dscattergood@customersystems.com](mailto:dscattergood@customersystems.com).