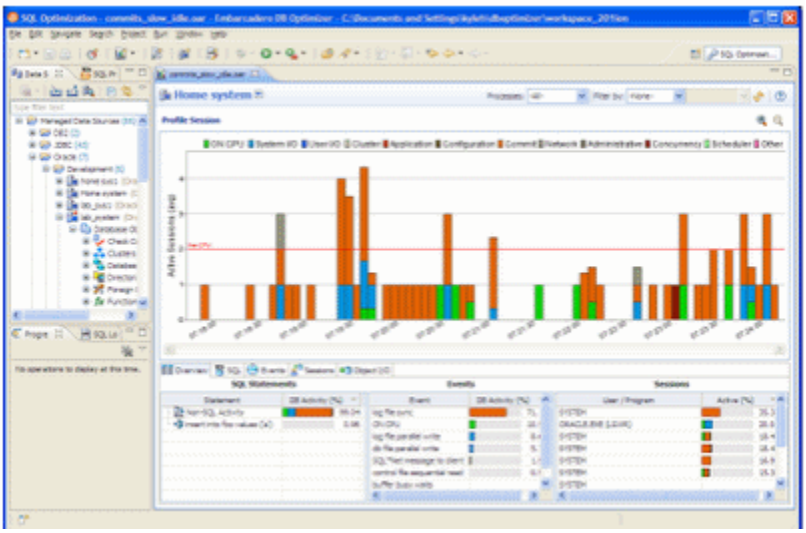
The Database is Hanging or the Application has Problems

I wonder if you can imagine, or have had the experience of the application guys calling with anger and panic in their voices saying, "The database is so slow, you've got to speed it up."

What's your first reaction? What tools do you use? How long does it take to figure out what's going on?

Let's take a look at how it would work with SQL Query Tuner.



We can clearly see that the database is not bottlenecked and there must be a problem on the application.

Why do we think it's the application and not the database? The database is showing plenty of free CPU in the load chart, the largest chart, on the top in the image above. In the load chart, there is a horizontal red line. The red line represents the number of CPUs on the system, which in this case is two CPUs. The CPU line is rarely crossed by bars which represent the load on the database, measured in average number of sessions. The session activity is averaged over five samples over five seconds, thus bars are five seconds wide. The bars above fall mostly about one average active session and the bars are rarely green. Green represents CPU load. Any other color bar indicates a sessions waiting. The main wait in this case is orange, which is log file sync, waits for commits. Why is the database more or less idle and why are most of the waits we do see for "commit"? When we look at the code coming to the database we see something like this:

```
insert into foo values ('a'); commit;
insert into foo values ('a'); commit;
insert into foo values ('a'); commit;
insert into foo values ('a'); commit;
insert into foo values ('a'); commit;
insert into foo values ('a'); commit;
insert into foo values ('a'); commit;
```

Doing single row inserts and committing after each is very inefficient. There is a lot of time wasted on network communication which is why the database is mainly idle. When the application thinks it's running full speed ahead, it is actually waiting mainly on network communication and commits. If we commit less and batch the work we send to the database, reducing network communications, we will run much more efficiently. Changing the code to:

```
begin

     for i in 1..1000 loop insert into foo values
     ('a');
     –commit;
     end loop; end;

/

commit;
```

improves the communication delay and now we get a fully loaded database but we run into database configuration issues.