



Tuning Oracle RAC databases with

empower!

Date:

August 2003

Overview:

Real Application Clusters (RAC) is Oracle's architecture for delivering a high-performance, always-available and scalable enterprise database platform. This paper discusses how you can use empower! from Linxcel Europe Ltd, to tune your RAC database.

Target Audience: DBMS technologists

About Linxcel Europe Ltd

Linxcel Europe Ltd is a privately held consultancy working to bring you the most cost effective, high performance Oracle environment, whether it's a simple traditional single instance solution or multi-instance, multi-node Real Application Cluster (RAC) operating on state of the art Intel/Linux blade technologies. The Linxcel philosophy and best practices for delivering a complete end-to-end Oracle system are set out in the following book:

High-Performance Oracle: Proven Methods for Achieving Optimum Performance and Availability

Author: Geoff Ingram

Publisher: John Wiley Inc, August 2002

ISBN: 0-471-22436-7

Founded by a professional technology management team with support from key customers, we aim to deliver a fast track to your preferred deployment. We deliver these solutions through:

- our consultancy services
- empower! for Oracle, our desktop tuning tool which includes extensive RAC-aware features
- our Corporate Advantage membership of the Oracle PartnerNetwork

For more information, contact consulting@linxcel.co.uk

A RAC Overview

At the heart of the RAC architecture is a single database shared co-operatively between multiple servers **at the same time**. As such, RAC is unique and fundamentally different from the traditional definition of clustering in the DBMS world. Traditional clustering usually means having a standby server available to take on the existing workload when a node fails, by remounting storage from the failed node onto the standby node, and migrating services on to it. In contrast, with RAC, all nodes are active at the same time. RAC brings additional performance capability into the equation, because all CPU and memory resources in your cluster as a whole are available to all workloads.

The RAC architecture itself is a key factor in delivering availability. All your client applications have access to the shared cluster database through any single node in your cluster. The cluster database is truly a shared resource. If a node dies unexpectedly, or you need to perform hardware maintenance on a node, then Oracle features such as Transparent Application Failover (TAF) are available to ensure that existing sessions are re-directed to any available node. Oracle's networking and load balancing technology directs new connections to the least heavily loaded nodes automatically.

Using RAC allows you to grow your computing resources in smaller incremental steps as demands increase. You "scale out" by purchasing additional lower cost servers and add them to the RAC configuration.

RAC uses a high-speed private network between cluster nodes (referred to as the "interconnect"), to ensure that data on all nodes in the cluster is managed as a seamless single entity. The interconnect network is a key component of "global cache fusion", a term used to describe the process by which Oracle keeps the individual database memory caches (SGAs) on each node in sync, cluster-wide.

Oracle Tuning Overview

The Oracle RDBMS has always facilitated performance tuning and management by presenting key performance metrics through database views. Viewing these metrics and how they change over time is usually the first step in identifying the root cause of a performance problem. These metrics are available through the V\$... performance views in the Oracle data dictionary, often referred to as the dynamic performance views. Since Oracle9i, cluster-aware equivalents of the dynamic performance views have been made available.

The most often-used system performance metrics fit broadly into two categories – statistics and events. Statistics are raw performance counters such as CPU used, and blocks read and written. Wait events on the other hand, indicate that sessions are queuing up to be serviced. As such, wait events may only manifest themselves on a busy system (often a production database) when end-user SQL statements wait for resources to become available.

A simple example can be used to demonstrate the difference between statistics and events (often referred to as wait events, or simply waits). When a SQL query causes a full table scan to occur, the evidence for the scan will be apparent by looking for an increase in the statistic “table scan rows gotten”. If you’re the only user on a development database, then it’s possible your session experiences no waits for resources to perform the I/O required by the scan. On the other hand, on a fully loaded production database where potentially multiple users may be running resource intensive SQL statements simultaneously, then your session may wait for a “db file scattered read” event, while Oracle completes other I/O before servicing yours. In both cases (given identical data and SQL), the statistic “table scan rows gotten” will have the same value for development and production. But in this example, wait events only show up for production.

Some Oracle performance experts recommend the use of waits in preference to statistics. This example should make it clear that you need facilities to measure and monitor both statistics **and** events to get a true picture of performance, depending on the circumstances. Also, tuning requirements may differ between your development and production environments. In development, performance management may involve a small number of developers tuning individual statements for maximum efficiency, whereas in production, performance management may involve managing a large end-user community competing for limited hardware resources. Of course, if your statements are as individually efficient as possible, then you have a much higher chance of avoiding resource contention issues in production.

Whatever the environment, for a RAC configuration, you need the ability to view statistics and events, and any other relevant performance metrics, for all instances in the cluster database at the same time.

The RAC Performance Challenge

To leverage the full benefits of RAC, your tools need to be RAC aware. When you need to tune and manage the performance of RAC databases, your traditional tools will struggle unless they are RAC aware.

RAC awareness means providing the ability to view performance of all instances in the cluster, and the cluster as a whole, all at the same time. If you can't do this, then whenever you see a cluster database performance problem, you may find yourself picking an instance at random at which to start your investigation, and iterating through a set of checks on each one manually until you find the one where the problem originates. Keep in mind also that the instance where a problem is manifested may be caused by an underlying problem on another instance. This challenge isn't such an issue for a two-node cluster. But the more nodes in your cluster, the greater the challenge. As a general rule, deploying more nodes in the cluster can lead to efficiencies of scale – but if this comes at cost of manageability, then that leaves you in a dilemma.

The solution is to choose a performance management and tuning tool for RAC databases that solves the RAC Performance Challenge.

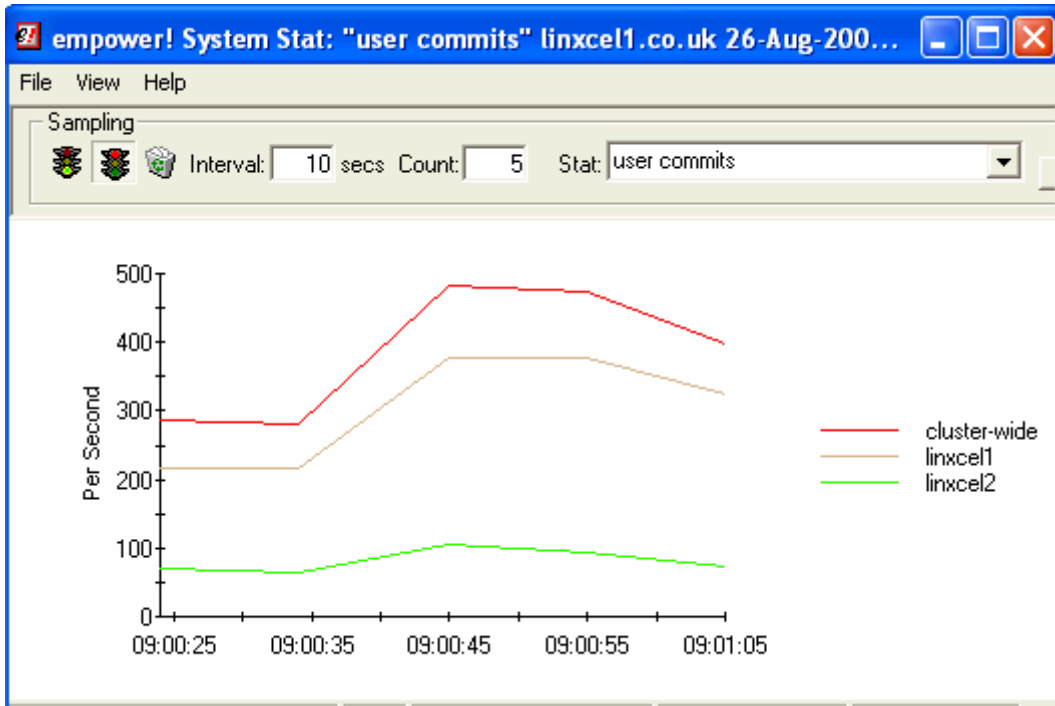
The Solution: “empower! for Oracle”

To solve your RAC performance challenges, Linxcel Europe Ltd have designed empower! for Oracle (referred to simply as empower!).

In brief, empower! is a lightweight, agent-free desktop PC performance tuning tool for Oracle databases, built from the ground up with RAC in mind. Installing in less than 1 minute, using industry-standard Microsoft Installer packaging technology, it can identify the cause of a RAC database problem in a short time, typically a few minutes, often sooner. This paper covers just a selection of the RAC-specific features in empower! – keep in mind that empower! is equally at home tuning a non-RAC configuration and contains many other features, such as client-side Tkprof and STATSPACK execution.

Measuring and Presenting Statistics

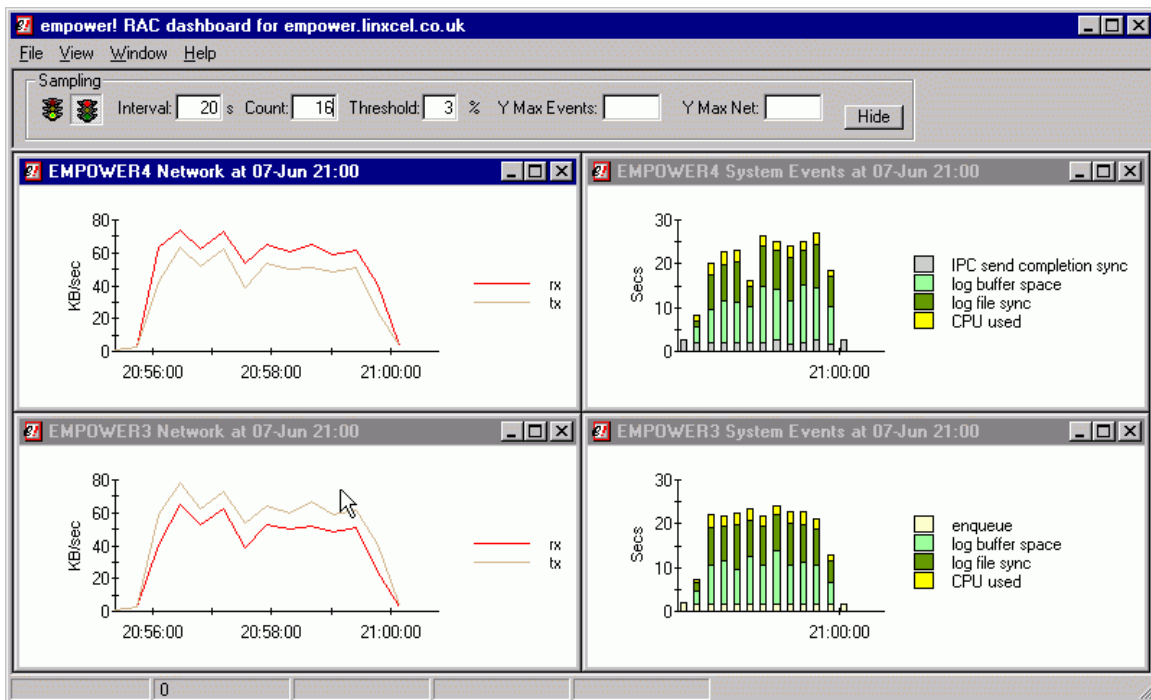
Through its Statistics Profile feature, empower! enables a user to monitor in real-time any of Oracle's ~250 system statistics, presented in a sliding window with time on the X axis. The statistic, chosen from a list, is displayed for each instance and the cluster as a whole. The number of samples display, and the interval between samples, is fully configurable. The example below shows the statistic "user commits" for two instances, linxcel1 and linxcel2, and the cluster-wide value, which is the sum of the instance values. You can display multiple charts, to monitor different statistics simultaneously. Charts can be printed and saved in JPEG format.



Measuring and Presenting Wait Events

Through its Events Profile feature, empower! enables a user to monitor in real-time the wait events for each cluster instance, side-by-side. The example below shows wait events for 2 cluster database instances, EMPOWER3 and EMPOWER4 in a 2-node cluster during an Oracle Import. A huge amount of performance information is available at glance. For example, the stacked bar charts show that in each 20-second interval that a bar represents, a few seconds are spent on CPU time, but most time is spent waiting for resources to become available.

Note: empower! **does not show idle events that indicate the system is waiting for work to be submitted.**



The specific event "log file sync" is associated with database commits, immediately identifying an opportunity for some tuning of commits. The "log buffer space" event indicates that instances are waiting for access to the redo log buffer, while its contents are written to disk. This indicates an opportunity for tuning of redo activity.

Because the Y-axes are scaled the same for all instances present in the cluster, the presence of bars of approximately the same height and colors for each instance indicate that the workload is of the same type for each instance and spread evenly across the nodes. If the bars are radically different heights for each instance, or event wait types differ significantly (indicated visually by different colors), then you have an immediate visual clue as to what's happening with cluster-wide performance. If you use Event Profiles a lot, you'll soon get a feeling for performance at a glance, based on bar heights and colors.

Note: for RAC systems on Linux, empower! can also display cluster-interconnect traffic as shown. Optionally, you can drill down and display Event Profiles for selected sessions in any cluster instance.

RACPACK and STATSPACK

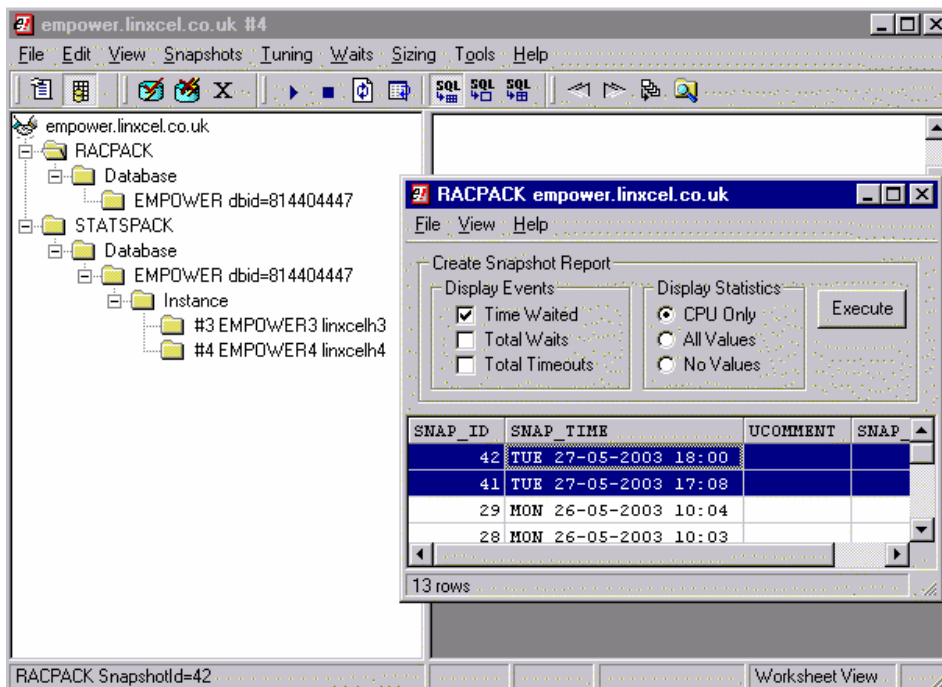
The collection and presentation of the statistics and events information discussed previously is aimed at addressing Oracle DBMS performance problems in real-time. This is possible because of the efficiency afforded by in-memory processing of the sampled information and presentation on the desktop PC, avoiding unnecessary workloads on the database server being monitored.

Oracle's STATSPACK is an essential part of the DBAs tuning kitbag, used extensively to capture system performance snapshots (including statistics and events) in the database, and to display reports comparing snapshots after the event. Typically, DBAs configure database jobs to schedule snapshots on a regular basis. Due to the performance overhead of taking a snapshot, the interval between snapshots is typically a few minutes at least. As such, STATSPACK provides lower granularity of information compared to the real-time monitoring discussed previously.

STATSPACK snapshots are captured for a single database instance at a time. If you want to use the facilities for a RAC database, you need to co-ordinate the simultaneous capture for each instance manually and design custom-reports to perform the comparisons.

Note: empower! enables you to take STATSPACK snapshots and compare them via a user-friendly graphical user interface that doesn't require a database server logon, or manual execution of SQL scripts on the database server.

To facilitate **cluster-wide** performance comparison, empower! introduces RACPACK. RACPACK is an add-on for STATSPACK that enables you to take snapshots of events and statistics performance metrics across the cluster as a whole, and compare them afterwards. This is essential for true understanding of RAC performance.



The previous screen shows two snapshots (ID 41 and 42) taken across a 2-node cluster shown in the Explorer tree on the left. The two instances and hosts for which STATSPACK samples are available are clearly shown under the STATSPACK node. Note that, unlike the STATSPACK node (which shows instance specific information) RACPACK doesn't include instance specific information in the tree because it samples cluster-wide – as a result, the RACPACK node includes only the database name. The power of RACPACK comes when presenting results, at which time instance specific information is shown.

In the interval between the snapshots, a 250MB import has been loaded locally into different schemas on each node. The RACPACK report looks like this:

N	SNA1	V3	V4
log buffer space	E	19121	18941
log file sync	E	15900	15812
global cache null to x	E	4706	4134
CPU used by this session	S	4300	5754
global cache busy	E	1860	1536
control file parallel write	E	570	574
global cache open x	E	529	472

RACPACK reports show the time spent in each instance for all system events (and optionally statistics) across the sample period. The more instances present at sample time, the more columns displayed. In this example for a 2-node cluster, instances with ID 3 and 4 only are present, so the time spent (in 1/100ths of a second) waiting on the metric in column N is shown in columns V3 and V4 respectively. For example, the time spent waiting on the "global cache null to x" event during the snapshot interval is 47.06 seconds for instance V3, and 41.34 seconds for instance V4.

In a 4-node cluster you would expect to see columns V1, V2, V3, V4. Columns only appear for instances that were up and running at the time of the cluster-wide snapshot. So if one node is missing from the report, then you know that node was down at the time of the sample.

In this example, the metrics themselves show that most time was spent loading data (due to the "log buffer space" and "log file sync" wait events). However, the effect of cache fusion on this particular two node Import is significant as demonstrated by the "global cache ..." events. Overall, the workload is distributed evenly across both nodes.

Note: RACPACK reports are provided through a PL/SQL procedural interface, so you can take snapshots and display reports via any PL/SQL compatible tool that supports cursor references.

Execution Plan and Auto Trace facilities

While you're developing SQL and PL/SQL you need to see extensive performance metrics on the statements as you run them. This ensures they are executing as efficiently as possible before you deploy them. Most DBAs will be aware that Oracle provides basic information using the SQL*Plus AUTOTRACE facility for DML statements. These facilities are taken many stages further by empower!. The following trace and explain facilities are available for each statement (including DDL) you execute in the worksheet:

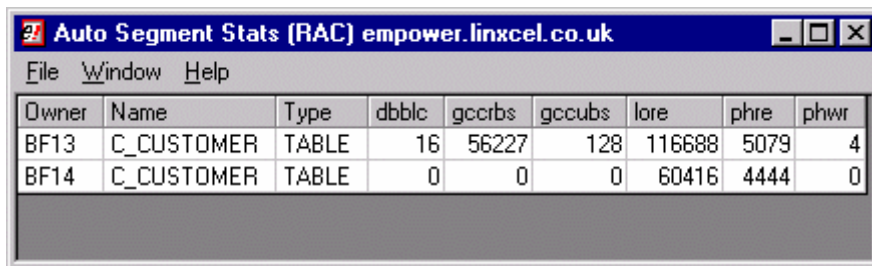
- All Session Statistics
- All Session Events
- Explain Plans - including 9.2 filter and access predicate information
- Execution Plans from V\$SQL_PLAN
- Segment Statistics - captured cluster-wide for RAC configurations

Whereas SQL statement plans can tell you the cost of each stage of execution (either predicted cost via EXPLAIN PLAN or actual cost via V\$SQL_PLAN), you need Oracle9i Segment Statistics to see the I/O (in blocks) that occurs on each object in your SQL statement during execution.

When you execute a query in the empower! worksheet with Auto Segment Stats enabled, empower! displays actual I/O statistics accumulated on all objects in the statement during the execution period. If the database is a RAC database, then empower! will show the origin of the blocks returned during execution if they come from a remote cache in another instance in the cluster.

The Segment Statistics obtained by running the SQL below show that many blocks are being fetched from other instances in the cluster to satisfy the BF13.C_CUSTOMER part of the query. The evidence is provided by the acronym gccbs, which stands for the statistic "global cache cr blocks served". The BF14.C_CUSTOMER rows on the other hand are being obtained locally, identified by phyr "physical reads" and lore "logical reads". In a RAC database you may well find that re-executing the same SQL immediately will show lower levels of global cache activity, because the blocks are now available in the local cache after the first run.

```
select count(*) from bf14.c_customer  
union all  
select count(*) from bf13.c_customer;
```



Owner	Name	Type	dbblc	gccrbs	gccubs	lore	phre	phwr
BF13	C_CUSTOMER	TABLE	16	56227	128	116688	5079	4
BF14	C_CUSTOMER	TABLE	0	0	0	60416	4444	0

Note: in the real empower! display a ToolTip tells you the expanded acronym name when you hold the cursor over a column in the Segment Stats grid.

Other RAC Tuning Facilities

This document has provided a taste of just some of the RAC tuning facilities in empower!. There are many others, shown below, and more are planned.

- Global Cache Statistics: current table or index block distribution in the global cache, by instance.
- RAC Healthcheck (Linux only): an extensive report on your Oracle/OS/Hardware configuration.

Many snapshots of cluster-wide performance metrics are available, some of them taken over a user-configurable (default 10 secs) snapshot interval, and presented in a spreadsheet-style view:

- Active Sessions: all currently active sessions, for each instance
- All Sessions: all current sessions
- Datafile I/O: read and write statistics for each datafile, by instance
- Global Cache Statistics: key cache fusion statistics, displayed by user session.
- Global Cache Performance: key cache fusion performance metrics, displayed by instance.
- Session Events: statistics by user session
- Session Waits: all waiting sessions, and the event waited for
- Session Statistics: statistics by user session
- Transactions: all currently active transactions, by session and instance

Many more reports are available for non-RAC configurations.

Client-side Tkprof

Like STATSPACK, Oracle's Tkprof server command line utility is another essential tool in the DBA's performance management kitbag.

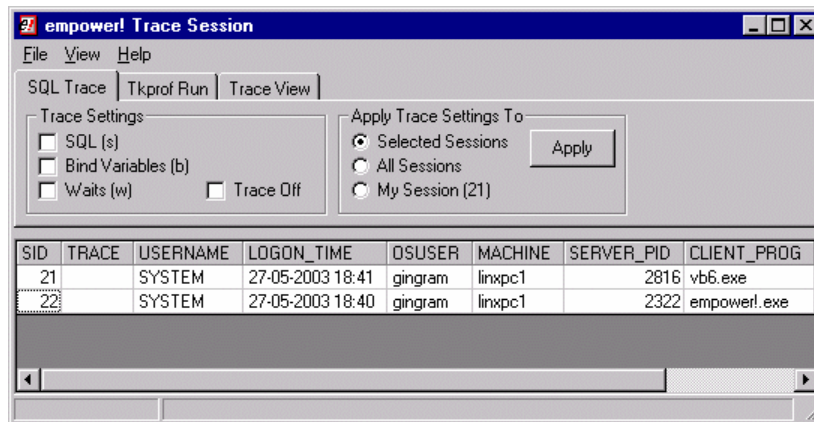
Traditionally, you need a database server logon in order to process and display trace files with Tkprof, because the trace files are generated on the database server. And as Tkprof is a command line tool, it can be difficult to remember the command line options. It's also hard to remember the trace settings required to turn on advanced features such as event wait and bind variable tracing.

With empower! you can enable and disable trace of various levels for your session or others, process the resulting trace on the server with Tkprof and display the resulting processed trace or raw output, all from within the empower! console on the PC desktop client – **no server logon is required**. This is a boon for developers who may not have a database server logon and need to rely on the DBA to generate profiled output for them, often resulting in delays. The tabs on the Trace Session screen follow the order in which processing takes place:

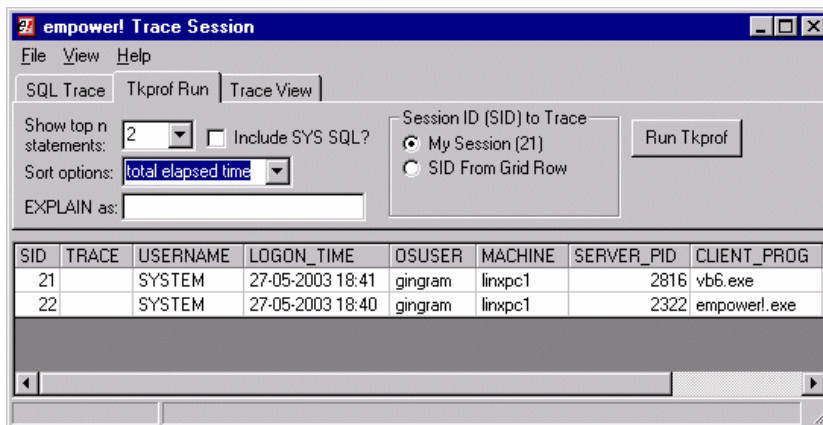
1. Enable trace generation
2. Process trace output
3. Display processed or raw trace output

Tkprof execution steps

To enable trace generation, empower! presents a list of current database sessions and trace options. You select one or more sessions to trace, and the trace options you want to enable which include regular SQL trace, trace with bind variables, and trace with wait events.



Next, you choose the options used to process the generated trace file. Often, DBAs and developers don't use these because they are hard to remember, resulting in a larger-than-required output file. In the example, only the top two statements ordered by elapsed time will be shown in the processed output.



The final step is to display either the processed or raw trace. For large trace files, or trace you want to search, you can optionally use Microsoft WordPad instead of displaying results in a built-in empower! form.

