DB2 UDB Performance Expert for Multiplatforms

A Usage Guide

- DB2 Performance Expert overview
- DB2 Performance Expert configuration considerations
- Usage scenarios

Nagraj Alur
Rohini Balaji
Marcia Miskimen
Doreen Stolz-Hofmann

ibm.com/redbooks

Redbooks

International Technical Support Organization

DB2 UDB Performance Expert for Multiplatforms A Usage Guide

June 2003

SG24-6436-00

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (June 2003)

This edition applies to Version 1, Release 1, Modification 7 of DB2 Performance Expert for Multiplatforms (product number 5724-B92).

Contents

Figures

Tables

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

	Distributed Relational Database Architecture™	IBM®
	DB2 Connect™	OS/390®
Redbooks (logo) 	DB2 Universal Database™	Perform™
ibm.com®	DB2®	Redbooks™
z/OS®	DRDA®	S/390®
AIX®		WebSphere®

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Preface

This IBM Redbook will help database administrators (DBAs) install, configure and exploit the new DB2® Performance Expert for Multiplatforms tool in their IT environment.

This book is organized as follows:

- **Chapter 1** provides an overview of DB2 performance management and describes the typical tasks performed by the DBA to ensure that their DB2 environment is meeting service level objectives.
- **Chapter 2** provides an overview of the architecture of DB2 Performance Expert for Multiplatforms and describes its main features and key functions.
- **Chapter 3** describes key considerations in configuring DB2 Performance Expert for Multiplatforms and provides recommendations for optimal use.
- **Chapter 4** discusses some typical problem scenarios encountered by DBAs and describes how the DB2 Performance Expert for Multiplatforms tool can be used to diagnose and resolve the performance problem.
- **Appendix A** describes the applications used in the problem scenarios.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Nagraj Alur is a Project Leader with the IBM International Technical Support Organization, San Jose Center. He has more than 28 years of experience in DBMSs and has been a programmer, systems analyst, project leader, consultant, and researcher. His areas of expertise include DBMSs, data warehousing, distributed systems management, and database performance, as well as client/server and Internet computing. He has written extensively on these subjects and has taught classes and presented at conferences all around the world. Before joining the ITSO in November 2001, he was on a two-year assignment from the Software Group to the IBM Almaden Research Center, where he worked on Data Links solutions and an eSourcing prototype.

Rohini Balaji is an Advisory IT Specialist from IBM India, Bangalore. She has seven years of experience in the IT industry and has worked at IBM for two years. Her areas of expertise include Java™, J2EE, WebSphere® and DB2.

Marcia Miskimen is a Software Engineer in the USA at IBM Software Group's Silicon Valley Lab, where she is currently a Data Management Tools Specialist. She has over 20 years of experience in the IT industry, including ten years as an IT Specialist in IBM Global Services. Her areas of expertise and interest include the application development life cycle, software testing, and tools of all sorts. She has a Bachelor of Science in Business Administration from Ohio State University.

Doreen Stolz-Hofmannis an IT Specialist with the IBM Software Group in Germany. She has over eight years of experience in managing very large database systems, and has been working with DB2 EEE and Data Warehousing for the past three years. She holds a degree in Informatik from the Technical University of Dresden. Her areas of expertise include design and migration of databases, performance management, and backup and restore management.

Thanks to the following people for their contributions to this project:

Emma Jacobs
Bart Steegmans
International Technical Support Organization, San Jose Center

Norbert Jenninger
Marian Steinberg
Manfred Werner
IBM Germany

Alice Ma
Mary Petras
Bryan Smith
Maria Weinerth
IBM USA

Tomoko Tanaka
IBM Japan

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. QXXE Building 80-E2
  650 Harry Road
  San Jose, California 95120-6099

1

Introduction to DB2 performance management

In this chapter, we provide a general overview of DB2 performance management concepts. We also describe the high-level tasks that a DBA typically performs to ensure that his/her DB2 environment is performing adequately and meeting previously agreed to service level objectives.

The topics covered are:

- Performance management
- Types of monitoring
- DB2 performance information
- DBA expectations

1.1 Performance management

DB2 UDB environments range from stand-alone systems to complex combinations of database servers and clients running on multiple platforms. Critical to all these environments is the achievement of adequate performance to meet business requirements. Performance is typically measured in terms of response time, throughput, and availability.

The performance of any system is dependent upon many factors, including system hardware and software configuration, number of concurrent users, and application workload.

Performance management is a complex issue and can be defined as modifying the system and application environment in order to satisfy previously defined performance objectives.

These performance objectives must be:

- **Realistic** in that they should be achievable given the current state of the technology available. For example, setting sub-second response times to process millions of rows of data is not achievable.
- **Reasonable** in that while the technology may be available, the business processes may not require stringent performance demands. For example, demanding sub-second response times for analytic reports that need to be studied and analyzed in detail before making a business decision could be considered unreasonable.
- **Quantifiable** in that the objectives must use quantitative metrics (numbers, ratios, percentages) instead of qualitative metrics (such as very good, average, etc.). An example of a quantitative metrics could be that 95% of a particular transaction time must have sub-second response time, while a qualitative metric could be that system availability should be very high.
- **Measurable** in that one has to be able to measure the performance in order to determine conformance or non-conformance with performance objectives. Units of measurement include response time for a given workload, transactions per second, I/O operations, CPU use, or a combination of the above. Setting a performance objective of sub-second response times for a transaction is moot if there is no means of measuring to determine whether this objective is being met.

Without well defined performance objectives, performance is a hit or miss exercise, with no way of delivering on any service level agreements that may be negotiated with users.

Figure 1-1 highlights the performance management cycle.

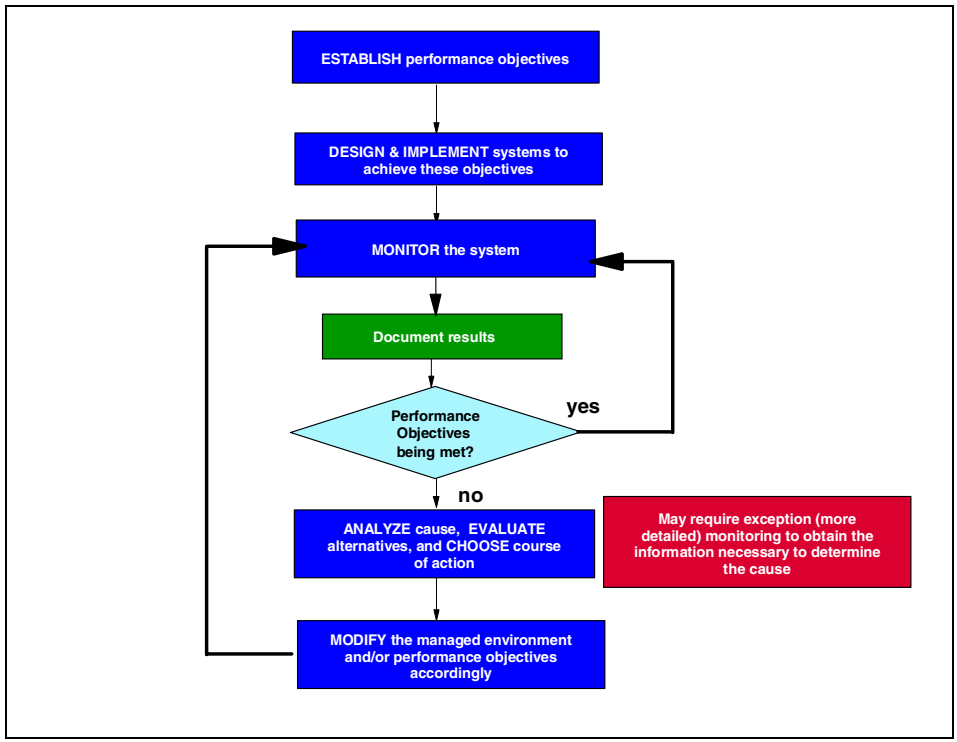


Figure 1-1 Performance management cycle

Performance management is an iterative process that involves constant monitoring to determine whether performance objectives are being met even as environments and workloads change over time. When performance objectives are not being met, then appropriate changes must be made to the hardware and/or software environment, as well as the performance objectives themselves, in order to ensure that the objectives will be met.

From a database perspective, performance problems can arise out of a combination of poor application and system design, inadequate CPU, memory disk, network resources, and suboptimal tuning of these resources.

Besides using monitoring to determine whether or not performance objectives are being met, monitoring is also used to:

- Assess the current workload of a system and track its changes over time for capacity planning purposes.
- Take a proactive approach to performance management by forestalling and resolving potential problems that could impede the achievement of performance objectives.
- React to unexpected problems by assisting in problem diagnosis.

Important: In order to meet performance objectives currently and in the future, the DBA needs to develop and execute an appropriate monitoring strategy capable of delivering the required quality of service.

1.2 Types of monitoring

There are typically three levels of monitoring available to a DBA, as follows:

- Routine monitoring
- Online/realtime event monitoring
- Exception monitoring

Each of these types of monitoring is described briefly in the following sections.

1.2.1 Routine monitoring

The objectives of this type of monitoring are to:

- Collect information about the workload and stress on the system during periods of normal and peak periods for capacity planning purposes as well as identifying potential performance problems down the road.
- Ascertain conformance of the system with performance objectives, and record any deviations.

The key to this type of monitoring is that it involves analyzing the information collected over a period of time (long history), and then taking corrective action, if required, to address performance objectives. In other words, there can be a significant delay between information collection and a corrective response. One example of the results of such monitoring is a realization that the number of transactions has been growing steadily at a 1% rate every week, which will necessitate an upgrade of the server in 12 months in order to continue to meet response time objectives.

Another characteristic of such monitoring is the critical need to minimize the overhead it introduces given its requirement to be running constantly or during peak periods.

In some literature, this type of monitoring is further subclassified into continuous monitoring (for normal loads) and periodic monitoring (for peak loads).

From a DB2 perspective, routine monitoring can help identify the root causes of potential performance problems, such as:

- Buffer pool size
- Dynamic cache size
- Heap sizes
- Locklist and maxlocks sizes
- Lock mode and isolation issues
- Disorganized table spaces
- Outdated runstats
- Long running SQL
- Log and table space utilization

1.2.2 Online/realtime event monitoring

The objective of this type of monitoring is to be on the lookout for specific events that may either identify a specific problem, or portend problems in the near to immediate future, in order to take prompt corrective action.

The key to this type of monitoring is that it involves looking for specific events, in a short interval of time (short history), that are known to degrade performance, and having the option to take prompt corrective action to rectify the problem. In other words, there probably needs to be a very short delay between information collection and a corrective response. One example of such an event is the occurrence of an excessive number of deadlocks in a short period of time, which need to be addressed promptly to ensure that business objectives are not being compromised.

Here too, the need to minimize the overhead of such monitoring is critical, given that most problems manifest themselves at peak loads.

From a DB2 perspective, event monitoring can help identify potential performance problems, such as:

- Deadlocks
- Long waits and timeouts
- Long running SQL

1.2.3 Exception monitoring

This type of monitoring is required when you discover or suspect a problem and need to identify its root cause in order to apply the appropriate corrective action to fix the problem.

Unlike routine and event monitoring, which are planned occurrences and are designed to have low overheads on the managed system, exception monitoring is driven by problem situations and may impose significant overhead on the managed system. An example of a need for exception monitoring is when the administrator receives a significant number of user complaints about degraded response times or inability to access the application. The administrator then needs to initiate a series of monitoring actions to hone in on the root cause of the problem. This typically involves coming up with a set of hypotheses that could account for the perceived behavior, and then systematically verifying each one in turn until the problem is diagnosed.

From a DB2 perspective, exception monitoring can apply to any of the items identified using routine and event monitoring.

1.3 DB2 performance information

DB2 UDB provides several tools and facilities for monitoring or analyzing database performance. This section describes them briefly, and provides guidelines for their use.

- **Snapshot Monitor:** captures performance information at periodic points of time. Details can be found in *IBM DB2 UDB V8 System Monitor Guide and Reference*, SC09-4847.
- **Event Monitor:** provides a summary of activity at the completion of events such as statement execution, transaction completion, or when an application disconnects. In DB2 V8, event monitors can write data to DB2 tables instead of files or pipes, thus enabling the information to be processed more easily using SQL. Details can be found in *IBM DB2 UDB V8 System Monitor Guide and Reference*, SC09-4847.
- **Explain Facility:** provides information about how DB2 will access the data in order to resolve the SQL statements. Details can be found in *IBM DB2 UDB V8 Administration Guide: Performance*, SC09-4821.
- **db2batch tool:** provides performance information (benchmarking tool). Details can be found in *IBM DB2 UDB V8 Administration Guide: Performance*, SC09-4821.
- **CLI/ODBC/JDBC Trace Facility:** traces all the function calls of DB2 CLI Driver, for problem determination and tuning applications using CLI, ODBC,

or SQLJ, or just to better understand what a third-party application is doing. Details can be found in *IBM DB2 UDB V8 Call Level Interface Guide and Reference Volume 1,* SC09-4849, and *IBM DB2 UDB V8 Call Level Interface Guide and Reference Volume 2,* SC09-4850.

- **db2diag.log:** prior to DB2 V8, this log had various degrees of diagnostic information recorded, depending upon the DIAGLEVEL database manager configuration parameter. There are five valid values for this parameter, ranging from zero to four. The default value is three, which captures all errors and warnings. A value of four captures informational messages a well. In Windows®, this file can be found in the SQLLIB/db2instance directory. In UNIX®, the db2diag.log is stored in the path specified in the DBM CFG parameter DIAGPATH if specified; the default installation path is INSTHOME/sqllib/db2dump.

  In DB2 V8, this log has been split into two:

  – db2diag.log and
  – db2admin log[1].

  The db2admin log will be used on all platforms for administration notification messages, while the db2diag.log will be targeted for use by troubleshooting personnel. The db2admin log will contain user-friendly messages that should enable the DBA to resolve minor issues. An API is available (db2AdminMsgWrite) that can be invoked to write messages to the db2admin log file; messages written by the API are distinguished from messages written by DB2. A new NOTIFYLEVEL dbm cfg parameter has been introduced that behaves like the DIAGLEVEL parameter, and provides granularity with respect to the severity of messages that are written to the db2admin log.

- **Design Advisor Wizard:** helps the administrator determine the best set of indexes for a given workload. A workload contains a set of weighted SQL statements that can include queries as well as updates. The wizard recommends which new indexes should be created, which existing indexes to keep, and which existing indexes to drop. The Design Advisor Wizard can be invoked from the DB2 Control Center, or using the db2advis utility. Details can be found in *IBM DB2 UDB Administration Guide: Performance*, SC09-4821.

- **Configuration Advisor Wizard:** helps the administrator tune the performance of the database by updating configuration parameters to match business requirements. The administrator specifies available memory and workload details, and the wizard recommends appropriate values for the database configuration parameters. The Configuration Advisor Wizard can be invoked from the DB2 Control Center.

---

[1] On UNIX platforms, the db2admin log is a text file called <*instance*> *.nfy* . On Windows, all administration notification messages are written to the Event Log. The errors can be written by DB2, the Health Monitor, the Capture and Apply programs, and user applications.

- **Health Monitor and the Health Center**

  DB2 V8 introduces two new features to help you monitor the health of DB2 systems:

  – Health Monitor
  – Health Center

  These tools add a management by exception capability to DB2 UDB by alerting the DBA to potential system health issues. This enables the DBA to address health issues before they become real problems that affect your system's performance. Details can be found in *IBM DB2 UDB V8 System Monitor Guide and Reference*, SC09-4847.

  The Health Monitor is a server-side tool that constantly monitors the health of the instance, even without user interaction. If the Health Monitor finds that a defined threshold has been exceeded (for example, the available log space is not sufficient), or if it detects an abnormal state for an object (for example, an instance is down), the Health Monitor will raise an alert. When an alert is raised, two things can occur:

  – Alert notifications can be sent by e-mail or to a pager address, allowing you to contact whoever is responsible for a system.
  – Preconfigured actions can be taken. For example, a script or a task can be run.

  The Health Center provides the graphical interface to the Health Monitor. The DBA can use it to configure the Health Monitor and to view the rolled up alert state of DB2 instances and database objects. The DBA can use the Health Monitor's drill-down capability to access details about current alerts and obtain a list of recommended actions on resolving the alert.

The following guidelines apply to the use of some of these tools:

- Choose the **Snapshot Monitor** or **Event Monitor** to gather data about DB2's operation, performance, and the applications using it. This data is maintained as DB2 runs, and can provide important performance and troubleshooting information.
- Choose the **Explain Facility** to analyze the access plan of an SQL statement or a group of SQL statements.
- Choose the **db2batch** tool to measure and analyze the performance of a set of SQL statements. Performance times can be returned along with Snapshot Monitor data for analysis. Explain information can be gathered for use by the Explain Facility.
- Choose the **CLI/ODBC/JDBC Trace Facility** to track activity between a CLI client and DB2. This facility can help pinpoint long running statements, and analyze the time spent in the client application, DB2, or the network.

- The **Design Advisor Wizard** and the **Configuration Advisor Wizard** should be run after significant changes to the workload have occurred or are anticipated. Given the potential for significant resource consumption by these wizards, these executions should be relegated to off-peak hours.
- Use the **Health Monitor** and **Health Center** to take a proactive approach to manage the DB2 environment by exploiting its management by exception capabilities.

Some of the monitoring tools include information collected by one or more of the other monitoring tools. For example, db2batch and the Event Monitor also display information collected by the Snapshot Monitor.

1.4 Database System Monitor information

The Snapshot Monitor and Event Monitor are the two primary tools for accessing Database System Monitor information.

The main types of information elements collected are:

- **Counter:** counts the number of times an activity occurs. Counter values increase during monitoring. Most counter elements are resettable. Some examples of counters include Deadlocks Detected, Number of Lock Escalations, and Maximum number of locks held (during this transaction).
- **Gauge:** indicates the current value for an item. Gauge values can go up and down, depending on database activity (for example, the number of locks held). Gauge elements are not resettable. Some examples of gauges include Lock Held, Total Lock List Memory in Use, and Connections Involved in Deadlock.
- **Water mark:** indicates the highest (maximum) or lowest (minimum) value an element has reached since monitoring was started. Water mark elements are not resettable. Some examples of water marks include Sort Private Heap High Water Mark, Catalog Cache High Water Mark, Package Cache High Water Mark, Maximum Number of Concurrent Connections, and Maximum Number of Coordinating Agents.
- **Information:** provides reference-type details of monitoring activities. This can include items such as partition names aliases and path details. Information elements are not resettable. Some examples of information include Lock Mode, Lock Status, and Lock Object Name.
- **Timestamp:** indicates the date and time that an activity took place by providing the number of seconds and microseconds that have elapsed since January 1, 1970. For the Snapshot and Event Monitors, the collection of timestamp elements is controlled by the TIMESTAMP switch. While this

switch is on by default, one should turn it off for performance reasons if CPU utilization on the database instance approaches 100%. Timestamp elements are not resettable. Some examples of timestamps include Lock Wait Start Timestamp, and Rollforward Timestamp.

- **Time:** returns the number of seconds and microseconds spent on an activity. For the Snapshot and Event Monitors, the collection of most time elements is controlled by the TIMESTAMP switch (new in DB2 UDB V8). While this switch is on by default, one should turn it off for performance reasons if CPU utilization on the database instance approaches 100%. Some timestamp elements are resettable.

1.5 DBA expectations

DBAs responsible for managing the performance of their database environment would appreciate having a single tool that provides the following capabilities:

- Access to all the monitoring and diagnostic support information such as the system monitor, explain feature, diagnostic logs, operating system, etc.
- Starter set database manager and database configuration parameters for their environment.
- Filtering capability to monitor only objects and items of interest, such as only specific databases/tablespaces/tables/applications, etc., and only look at specific attributes such as elapsed time, buffer hit ratios, etc.
- Recommendations for setting up routine and event monitoring.
- Troubleshooting guidance for common problems.
- Recommendations to resolve problems.
- Proactive approach to performance management, to forestall impending performance problems and provide guidance on resolving them before they occur.
- Automatic detection and resolution of common problems.
- Guidance with the identification and resolution of complex problems.

Note: While there is currently no single tool capable of delivering on these expectations, DB2 PE is being continuously enhanced to address these DBA requirements in future releases. In the meantime, the DBA has to conduct performance management using a collection of tools, best practices documentation, consultants and personal experience.

2

Overview of DB2 Performance Expert

In this chapter, we briefly describe the architecture and main components of DB2 Performance Expert for Multiplatforms; we also highlight the various performance metrics provided to the DBA for managing their DB2 environment.

The emphasis is on providing familiarity with the tool, rather than advising the DBA about DB2 performance considerations.

The reader is strongly urged to consult the DB2 Performance Expert for Multiplatforms documentation for detailed information on navigating its various functions and features.

The topics covered include:

- Introduction to DB2 Performance Expert
- DB2 PE for Multiplatforms environment structure
- Main features and functions

2.1 Introduction to DB2 Performance Expert

The DB2 Performance Expert (PE) is a new IBM supplied host-based and workstation-based performance analysis and tuning tool for both the z/OS® and multiplatform environments. Many installations have more than one DB2 system. At a minimum, you might have test, production and other systems; in some cases, there may also be a mix of z/OS and multiplatform environments. With DB2 PE, one can manage a heterogeneous mix of DB2 systems using a single end-user interface.

The main objective of DB2 PE is to simplify DB2 performance management. DB2 PE gives one the capability of monitoring applications, system statistics and system parameters using a single tool.

DB2 PE integrates performance monitoring, reporting, buffer pool analysis, and a Performance Warehouse function into a single tool. It optimizes the performance of IBM DB2 for z/OS and OS/390®, and of DB2 UDB by providing a comprehensive view of DB2 performance-related information. It also provides you with reports, analysis, and recommendations.

In general, DB2 PE includes the following advanced capabilities:

- Provides detailed analysis of key performance factors that let you control and tune the performance of DB2 and DB2 applications.
- Provides a realtime online monitor, a wide range of reports, expert analysis, and an explain feature to analyze and optimize SQL statements.
- Lets you simulate certain tuning actions of buffer pools before you change your system.
- Includes a Performance Warehouse function for storing performance data and analysis functions.
- Lets you monitor Database Connection Services (DCS) connections using Performance Expert Agent.

The following subsections provide a summary of all functions. The availability of these functions, however, varies, depending on whether you install DB2 Performance Expert for z/OS, DB2 Performance Expert for Multiplatforms, or one of the stand-alone products DB2 PM or Buffer Pool Analyzer.

2.1.1 Basic functions on all platforms

DB2 PE provides the following functions across all platforms:

- Analyzes and tunes the performance of DB2 and DB2 applications.
- Provides expert analysis, a realtime online monitor, a wide range of reports for analyzing and optimizing DB2 applications and SQL statements.
- Includes a Performance Warehouse for storing performance data and analysis tools.
- Defines and applies analysis functions (rules of thumb, queries) to identify performance bottlenecks.

2.1.2 Specific functions on z/OS

DB2 PE provides the following functions on z/OS:

- An explain feature.
- A Reporting Facility that presents detailed information about DB2 events involving CPU times, buffer pool usage, locking, I/O activity, and more.
- The ability to manage buffer pools more efficiently by providing information about current buffer pool behavior and simulating anticipated future behavior.
- Exception reports for common performance problems to help identify and quantify excessive CPU and elapsed time on a plan and package basis.

2.1.3 Specific functions on Multiplatforms

DB2 PE provides the following functions on Multiplatforms:

- A starter set of smart features that provide recommendations for system tuning to gain optimum throughput.
- A subset of the Reporting Facility functionality provided on z/OS.
- A subset of the Exception Reports functionality provided on z/OS.
- DB2 Buffer Pool Analyzer, which collects data and provides reports on related event activity, to obtain information on current buffer pool behavior. It can provide these reports in the form of tables, pie charts, and diagrams.
- Monitoring of DB2 Connect™ Gateways, including application- and system-related information.

Important: For a detailed description of the different capabilities and specific platform support, refer to the document *IBM DB2 Performance Expert for z/OS and Multiplatforms Monitoring Performance from the Workstation*, SC27-1645-03, and the corresponding Announcement Letters.

2.1.4 Main components of DB2 Performance Expert

The main components of the DB2 Performance Expert Tool are shown in Figure 2-1.

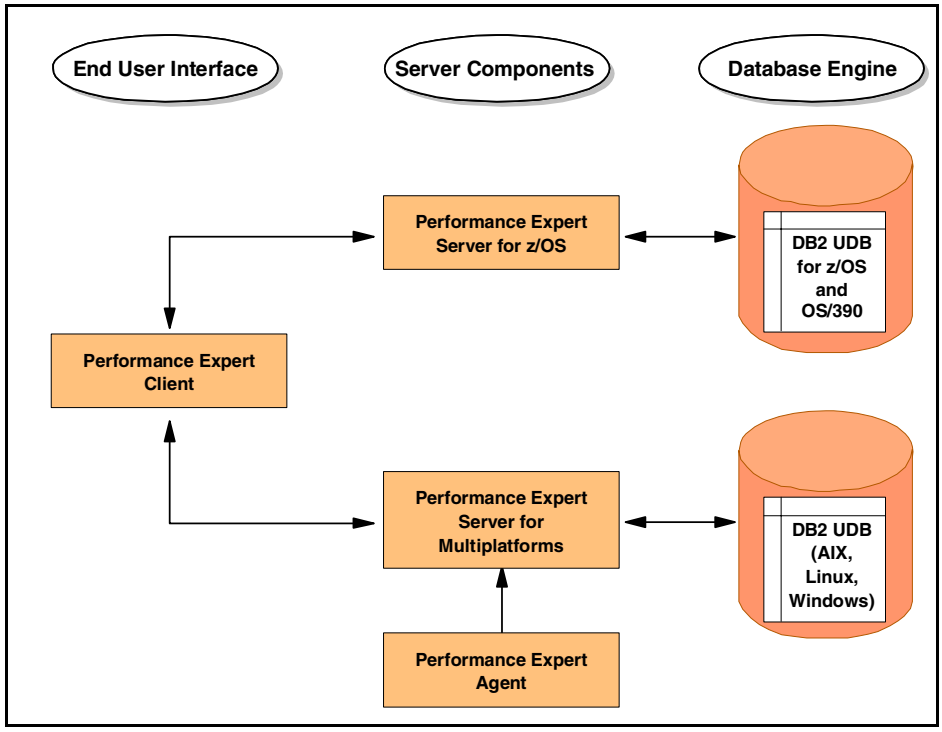


Figure 2-1 Main components of DB2 Performance Expert

A brief description of each of these components follows:

- Performance Expert Client designates the end user interface of DB2 PE of z/OS and Multiplatforms.
- Performance Expert Server for z/OS accesses DB2 UDB for Z/OS and OS/390.

- Performance Expert Server for Multiplatforms accesses DB2 UDB for Windows, UNIX and Linux (IA32).
- Performance Expert Agent monitors Database Connection Services (DCS) connections within the Distributed Relational Database Architecture™ (DRDA®). When Performance Expert Agent is installed on the system on which DCS connections are performed, it collects connection-related data, such as the status of a DCS connection. It also collects statistics about DB2 Connect activities. The collected data is then stored in the DB2PM database on a Performance Expert Server. Figure 2-2 provides an overview of how Performance Expert Agent is integrated into the system environment. The DB2PM database (also known as the DB2 Performance Expert Performance Database) stores performance data collected from the initiation of various traces and event monitors. DB2 PE provides various canned queries to report the content of this performance data.

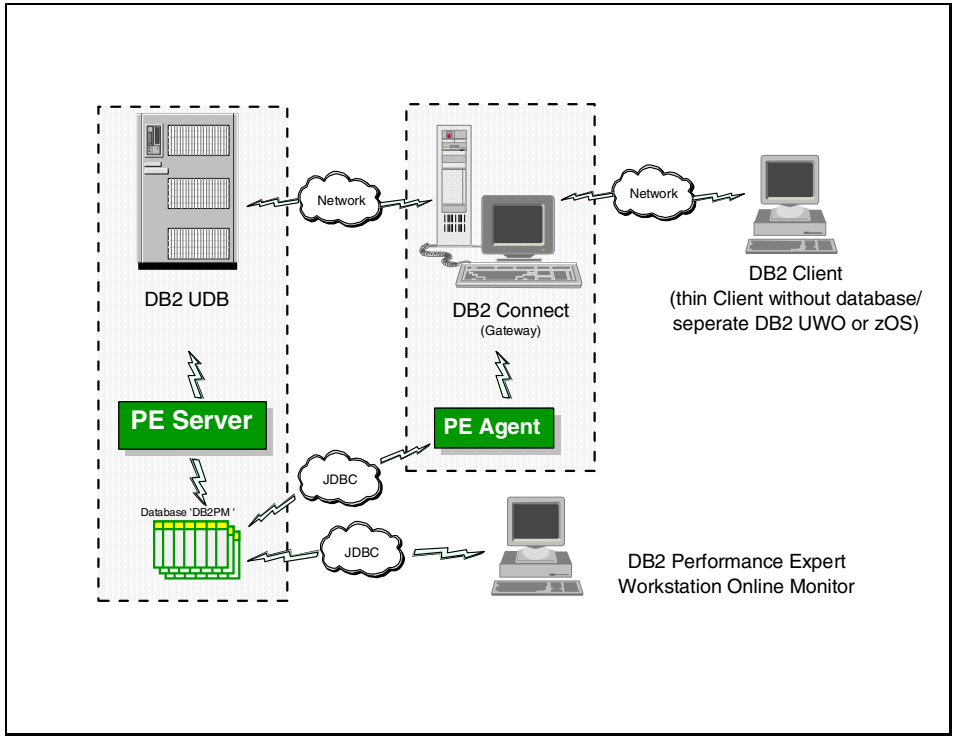


Figure 2-2 Performance Expert Agent and system environment

2.2 DB2 PE for Multiplatforms environment structure

Figure 2-3 describes the general environment structure of DB2 PE for Multiplatforms.

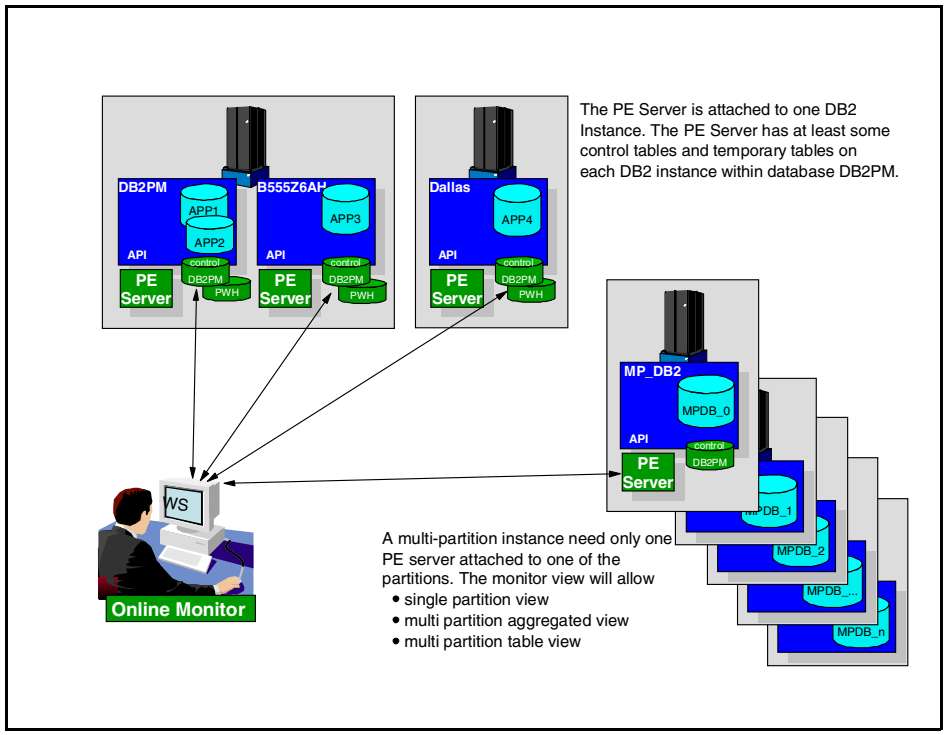


Figure 2-3 DB2 PE for Multiplatforms environment structure

The figure shows the DB2 PE Client monitoring two DB2 EE systems and one DB2 EEE system.

It should be noted that there is a DB2 PE Server and a DB2PM database associated with each DB2 instance, with a future goal of sharing a single DB2PM database across multiple DB2 instances.

Note: Performance Warehouse (PWH) tables are included in the DB2PM database.

Both the DB2 PE Client and the DB2 PE Server support Windows 2000, AIX®, Solaris, HP-UX and Linux (IA32) platforms.

2.3 Main features and functions

DB2 Performance Expert collects a wide variety of performance data in order to deliver its performance management functionality. This section is organized as follows:

1. **Kinds of data collected** describes the broad categories of data collected by DB2 PE and the menu items that provide access to such information.
2. **DB2 PE menu overview** describes the functionality of the main menu items such as Application Summary, Statistics Details, System Health, Applications in Lock Conflicts, Locking Conflicts, System Parameters, Performance Warehouse and Buffer Pool Analysis. Also described is the history mode capability. This is meant to provide an introduction to the functionality only; the reader should refer to the DB2 PE product documentation for complete details on the setting and navigating these capabilities.
3. **Data flow in DB2 PE** describes where persistent and transient performance data and other information is stored.
4. **DB2 PE menu items vis-a-vis types of monitoring** describes how the DBA could leverage DB2 PE functionality to perform monitoring activities such as routine monitoring, online/realtime monitoring and exception monitoring.

2.3.1 Kinds of data collected

Figure 2-4 describes where, how, and when the various kinds of performance data are collected and stored, and the mechanisms available in DB2 PE to display and report on it.

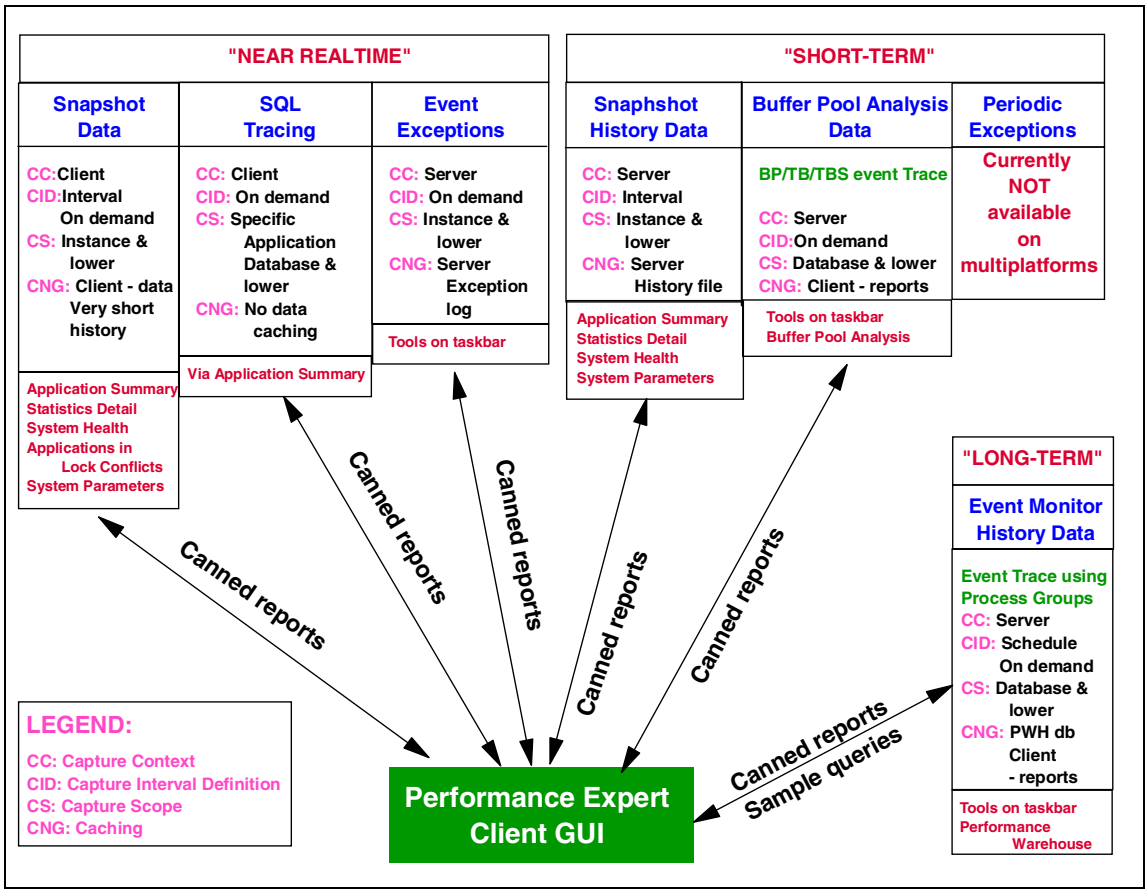


Figure 2-4 Categories of information collected by DB2 PE

The data collected by DB2 PE may be broadly categorized as follows:

- **"Near realtime"** data about the state of an activity at a point in time in the DB2 instance being monitored. This data is captured at user-defined intervals which tend to be measured in a few seconds (the default is six seconds), and it is therefore termed "near realtime". Figure 2-4 shows the menu items in the DB2 PE workstation GUI that show "near realtime" data. For example, "near realtime" snapshot data is accessible from the Application Summary, Statistics Detail, System Health, Application in Lock Conflicts, and System

Parameters tasks in the DB2 PE System Overview as shown in Figure 2-5 on page 21.

- **"Short-term"** snapshot history data refers to the storing of only the most recent (from a few minutes to a few hours in the recent past) performance-related data (output of the **`db2 get snapshot`** command), for projecting trends using graphs or for looking at the state of an activity at a given point in time in the recent past. Figure 2-4 shows the menu items in the DB2 PE workstation GUI that show "short-term" snapshot history data. For example, "short-term" snapshot history data is accessible from the Application Summary, Statistics Detail, System Health, and System Parameters tasks in the DB2 PE System Overview, as shown in Figure 2-5 on page 21.
- **"Long-term"** event monitor history data refers to the storing of all performance-related data (output of the CREATE EVENT MONITOR ... and SET EVENT MONITOR STATE ... SQL statements) for considerable periods of time, for conducting long-term capacity planning, as well as forestalling potential problems based on changes in the workload or available resources. Figure 2-4 shows the menu items in the DB2 PE workstation GUI that show "long-term" history information. For example, "long-term" event monitor history data that is stored in the Performance Warehouse is accessible from the Tools taskbar in the DB2 PE System Overview, as shown in Figure 2-5 on page 21.

Figure 2-4 summarizes the categories of data collected, and the avenues available in DB2 PE to interact with this data.

The description of the legend in Figure 2-4 is as follows:

- **Capture context** (CC) refers to the context in which the information requested is captured.
  - **Client** indicates that each PE Client can request this information, and that this information is collected by the PE Server on a PE Client by PE Client basis, that is, each PE Client request is processed independent of the other.
  - **Server** indicates that no matter which PE Client requests this information, it is collected only once at the PE Server level. This PE Server information is then available to all PE Clients requesting that information.
- **Capture Interval Definition** (CID) refers to the interval/frequency at which this information is collected.
  - **Interval** indicates that one can specify an interval (in seconds/minutes/hours) at which the information should be collected.
  - **On demand** specifies that one may request this information as and when desired through the click of a button.

  - **Schedule** specifies that the information can be requested on a scheduled basis, such as specific days and times of the week.

- **Capture scope** (CS) refers to the scope of the information that may be captured. For example, it may only be possible to capture at an instance level in some cases, while in others it would be possible to capture information at a database and tablespace level.
  - **Instance and lower** specifies that the scope is at an instance level and lower, and therefore includes databases, tablespaces and the like.
  - **Database and lower** specifies that the scope is a database, tablespace, etc.
- **Caching** (CNG) refers to any form of caching that is supported when information is captured or processed.
  - **Client - data** specifies that data from the captured information is cached in the PE Client, and that this caching occurs in memory only and is therefore volatile. In other words, when the PE Client is shut down and restarted, this information is lost.
  - **PWH db** specifies that the captured information is stored in the Performance Warehouse database, and therefore persistent.
  - **Client - reports** specifies that the reports generated from processing the captured information are cached in the PE Client, and that these reports are persistent, since they are stored as files on the PE Client.
  - **Server - history** specifies that “history mode” performance information is stored on the server in a pair of files that are used in wraparound fashion. While this information is persistent, it can be overwritten when wraparound occurs.
  - **Server - Exception log** specifies that event monitor output (only deadlocks are currently supported) is stored in a persistent log, which is accessible from the Tools menu on the taskbar.

2.3.2 DB2 PE menu overview

This section covers some of the functionality of the main menu items and history mode.

Note: For full details about these features, refer to *IBM DB2 Performance Expert for z/OS and Multiplatforms: Monitoring Performance from the Workstation*, SC27-1645.

The System Overview window of Performance Expert Client is shown in Figure 2-5.

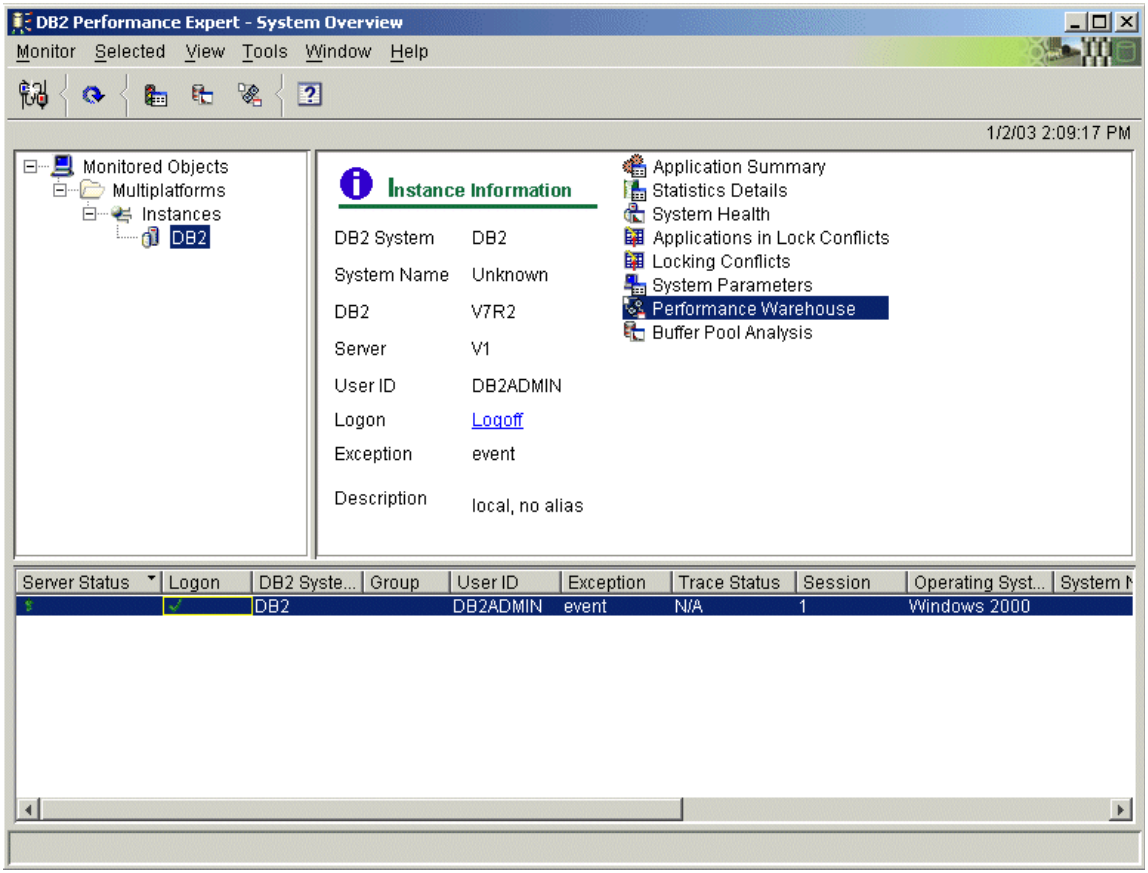


Figure 2-5 DB2 Performance Expert Client - System Overview

Each of these menu items is described briefly in the following sections.

History mode

This feature of DB2 PE enables one to specify the collection of *DB2 instance wide* performance data at user-defined intervals, which can then be viewed by activating the history slider on a number of monitoring windows, including Application Summary, Statistics Details, System Health, Applications in Lock Conflicts, and System Parameters windows.

At user-specified intervals, the **`db2 get snapshot`** command is issued by DB2 PE, and the output is written to a pair of history files that are used in wraparound

mode. When the history slider is activated on the appropriate monitoring function window, this snapshot data is retrieved from the history file and displayed on the DB2 PE Client. This capability provides an excellent opportunity to review online the state of the monitored system at a prior point in time, to assist with problem diagnosis in particular, and routine monitoring at peak intervals in general.

Important: The size of the history files, the frequency of snapshot data collection, and the workload on the monitored system dictate the amount of history information available for review using the history slider.

Setting history mode

To enable history data collection, select the DB2 instance of interest in the System Overview window and click **Selected -> Properties**, then select the **History** tab as shown in Figure 2-6.

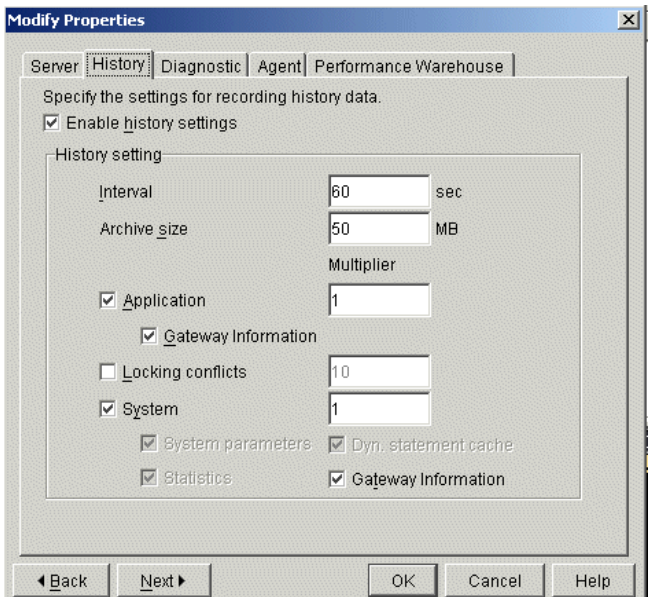


Figure 2-6 Setting History mode properties

The *Interval* specifies the frequency of snapshot data collection, while the *Archive size* specifies the size of the history file. The *Multiplier* setting controls how often the subcategory history snapshot data is collected.

The basic information is collected with every single snapshot, while the subcategories such as *Application*, *Locking conflicts*, and *System* are collected at less frequent intervals as determined by the *Multiplier*. For example, if the Interval setting is 60 seconds (the default), and you want to capture Application

information every five minutes, then the Multiplier value should be set to 5 for Application. This will result in every fifth snapshot sent to the history file containing information about applications in addition to the basic information.

Viewing snapshot history data

As mentioned earlier, snapshot history data can be viewed in a number of monitoring function windows. The following describes viewing history data in the Application Summary window.

Double-click the **Application Summary** menu item in the System Overview window (see Figure 2-5 on page 21) to view the history option as shown in Figure 2-7. Activate the history mode by clicking the history button.

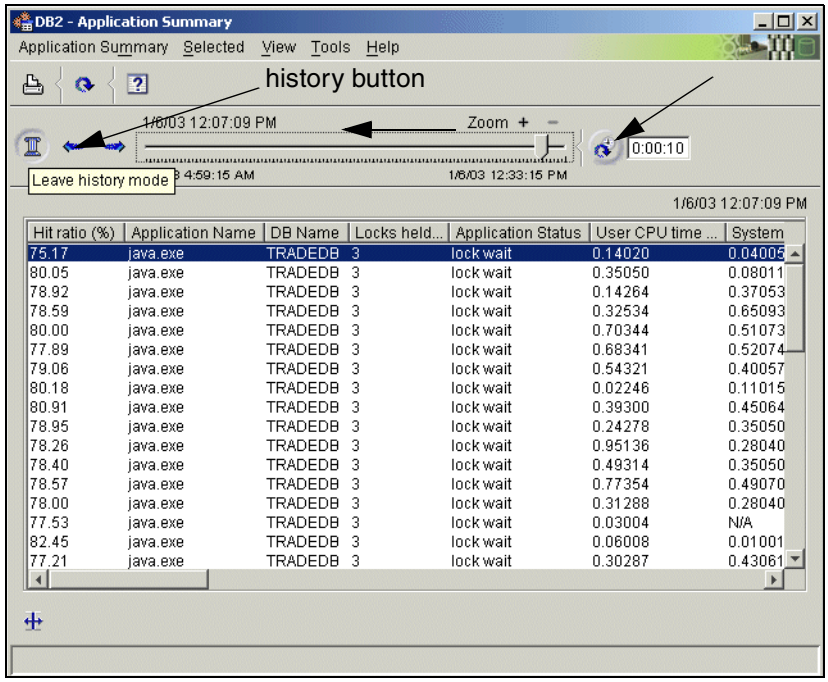


Figure 2-7 Application Summary in history mode

The history slider (highlighted with an arrow) is divided into units, where each unit represents the date and time when a snapshot was taken. The timestamp above the history slider indicates the date and time of the snapshot at the current slider position, which is 1/06/03, 12:07:09 p.m. in our example. Moving the cursor over the timestamp displays the time interval covered by the slider.

To view data related to a specific snapshot, click **View -> History Settings** as shown in Figure 2-7 on page 23 to display the History Settings window shown in Figure 2-8. You may set the specific date and time in this window, and click **OK** to view the corresponding snapshot data.

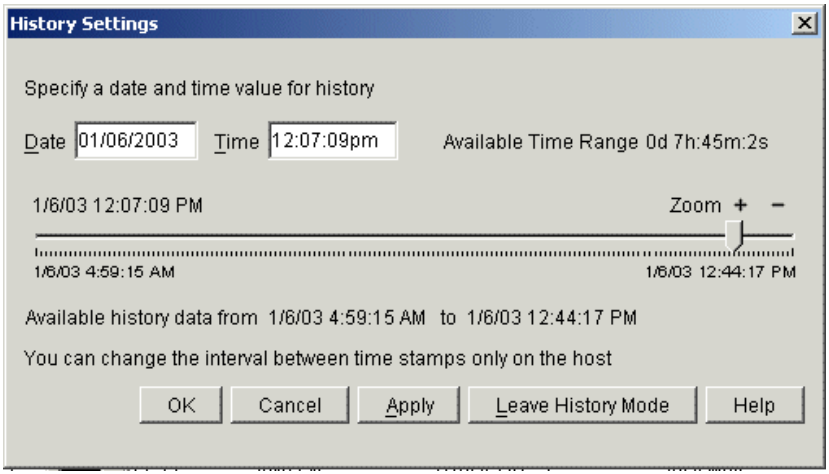


Figure 2-8 History slider interval and settings

As shown in Figure 2-7 on page 23, you may click the left arrow (<-) or right arrow (->) to view the previous or next snapshot history data.

Pausing history

If you disable history collection, no snapshots are collected and nothing new is written out to the history files.

When you next enable history, you will see a gap in the history mode slider bar indicating no history data collection for that period. Eventually, the wraparound history files will flush this gap as new performance data gets written. Such a gap can also occur if DB2 PE Server is shut down for some time and then restarted.

Important: There is sometimes a tendency to get confused between the history snapshot interval specified at the DB2 instance level (default of 60 seconds), and the auto refresh interval (default of 6 seconds) specified on each of the monitoring function windows.

The history slider allows you to step through snapshots collected at the history snapshot interval.

The auto refresh interval, on the other hand, applies to each monitoring function on a particular DB2 PE Client, and specifies the frequency at which the performance data displayed on the DB2 PE Client window is to be refreshed. At intervals specified by auto refresh (or on demand), a **`db2 get snapshot`** command is issued to collect snapshot data for presentation on the DB2 PE Client.

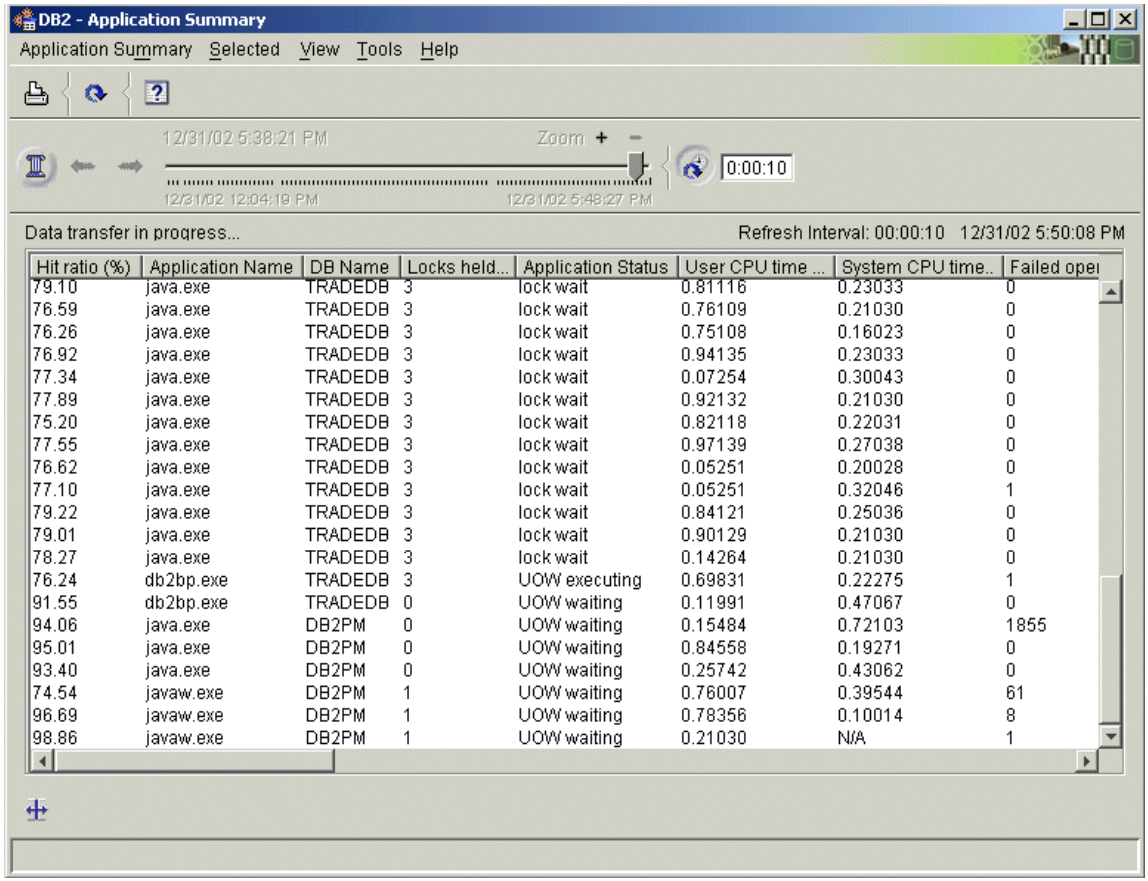
Therefore, attempts to synchronize the history slider snapshot data and the traverses interval snapshot data on monitoring function windows will invariably be fruitless in an active system.

Application Summary

The Application Summary monitoring function lists all the active applications connected to a DB2 instance. This information is collected on demand or at the auto refresh interval (default of 6 seconds) when DB2 PE issues a **`db2 get snapshot`** command.

The history slider can be activated to view active applications at a previous point in time when problems were reported.

The Application Summary monitoring function can be invoked by selecting the DB2 instance of interest, then double-clicking **Application Summary**, which displays the window shown in Figure 2-9.



Hit ratio (%)	Application Name	DB Name	Locks held...	Application Status	User CPU time ...	System CPU time..	Failed oper
79.10	java.exe	TRADEDB	3	lock wait	0.81116	0.23033	0
76.59	java.exe	TRADEDB	3	lock wait	0.76109	0.21030	0
76.26	java.exe	TRADEDB	3	lock wait	0.75108	0.16023	0
76.92	java.exe	TRADEDB	3	lock wait	0.94135	0.23033	0
77.34	java.exe	TRADEDB	3	lock wait	0.07254	0.30043	0
77.89	java.exe	TRADEDB	3	lock wait	0.92132	0.21030	0
75.20	java.exe	TRADEDB	3	lock wait	0.82118	0.22031	0
77.55	java.exe	TRADEDB	3	lock wait	0.97139	0.27038	0
76.62	java.exe	TRADEDB	3	lock wait	0.05251	0.20028	0
77.10	java.exe	TRADEDB	3	lock wait	0.05251	0.32046	1
79.22	java.exe	TRADEDB	3	lock wait	0.84121	0.25036	0
79.01	java.exe	TRADEDB	3	lock wait	0.90129	0.21030	0
78.27	java.exe	TRADEDB	3	lock wait	0.14264	0.21030	0
76.24	db2bp.exe	TRADEDB	3	UOW executing	0.69831	0.22275	1
91.55	db2bp.exe	TRADEDB	0	UOW waiting	0.11991	0.47067	0
94.06	java.exe	DB2PM	0	UOW waiting	0.15484	0.72103	1855
95.01	java.exe	DB2PM	0	UOW waiting	0.84558	0.19271	0
93.40	java.exe	DB2PM	0	UOW waiting	0.25742	0.43062	0
74.54	javaw.exe	DB2PM	1	UOW waiting	0.76007	0.39544	61
96.69	javaw.exe	DB2PM	1	UOW waiting	0.78356	0.10014	8
98.86	javaw.exe	DB2PM	1	UOW waiting	0.21030	N/A	1

Figure 2-9 Application Summary

More detailed information about a specific application in the list can be obtained by selecting that application and double-clicking it.

Figure 2-10 shows the SQL Statement and Package selection of the Application Details window for a particular application. Other options available for selection are Locks, Cache, Buffer pool, etc.

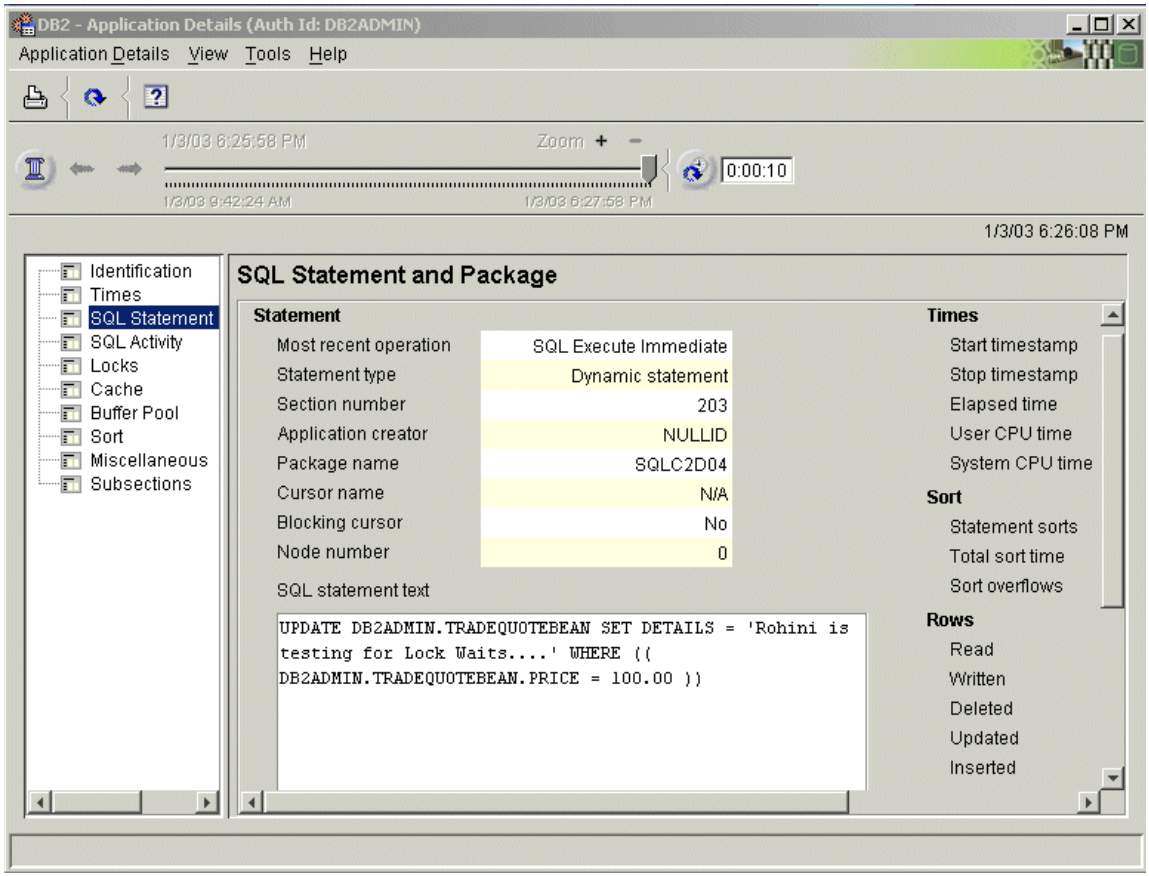


Figure 2-10 Application Summary - Application Details

Statistics Details

The Statistics Details monitoring function provides information about key DB2 performance data, including percentages and important DB2 thresholds. This information is collected on demand or at the auto refresh interval (default of 6 seconds) when DB2 PE issues a **`db2 get snapshot`** command.

The history slider can be activated to view Statistics Details information at a previous point in time when problems were reported.

The Statistics Details monitoring function can be invoked by selecting the DB2 instance of interest, then double-clicking **Statistics Details**, which displays the window shown in Figure 2-11.

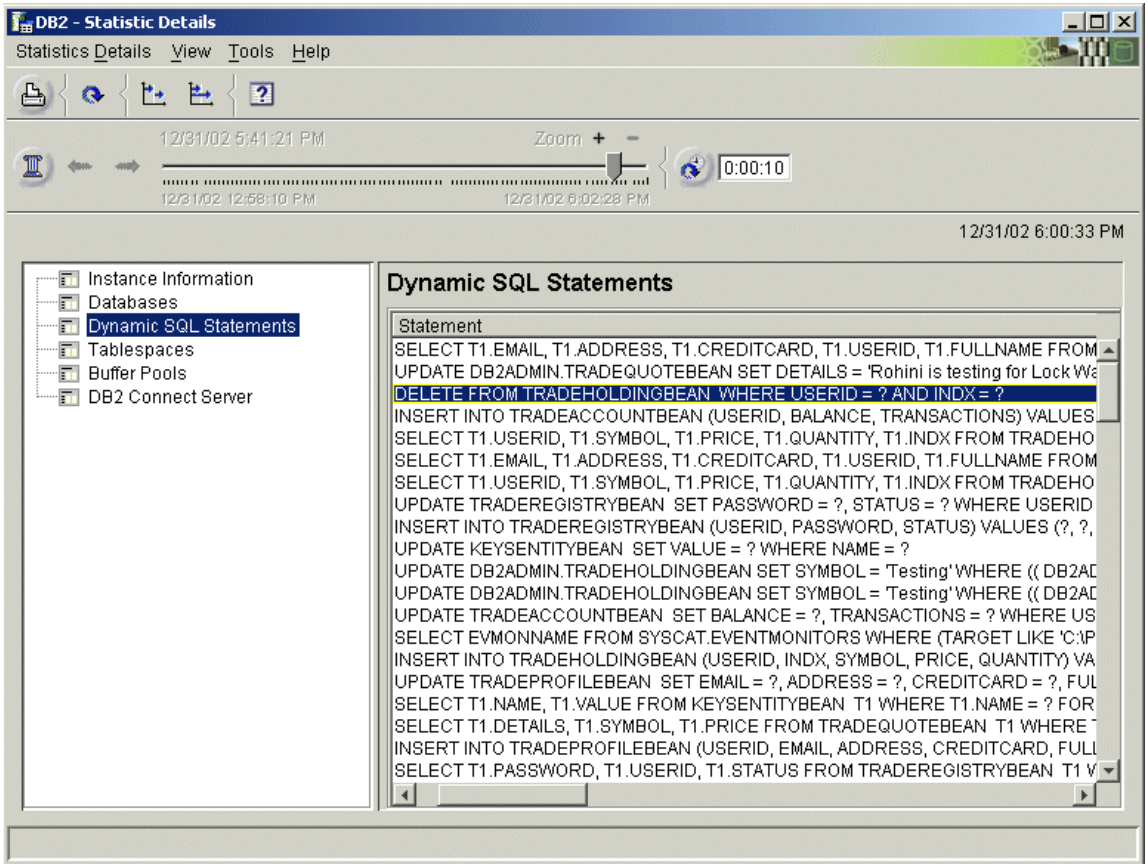


Figure 2-11 Statistic Details - Dynamic SQL Statements

Except for the Databases page, all pages provide statistics information at the DB2-instance level. The Database page and its sub-windows provide statistics information at the database level.

Figure 2-12 and Figure 2-13 on page 30 show statistics information at the database level, TRADEDB in our example.

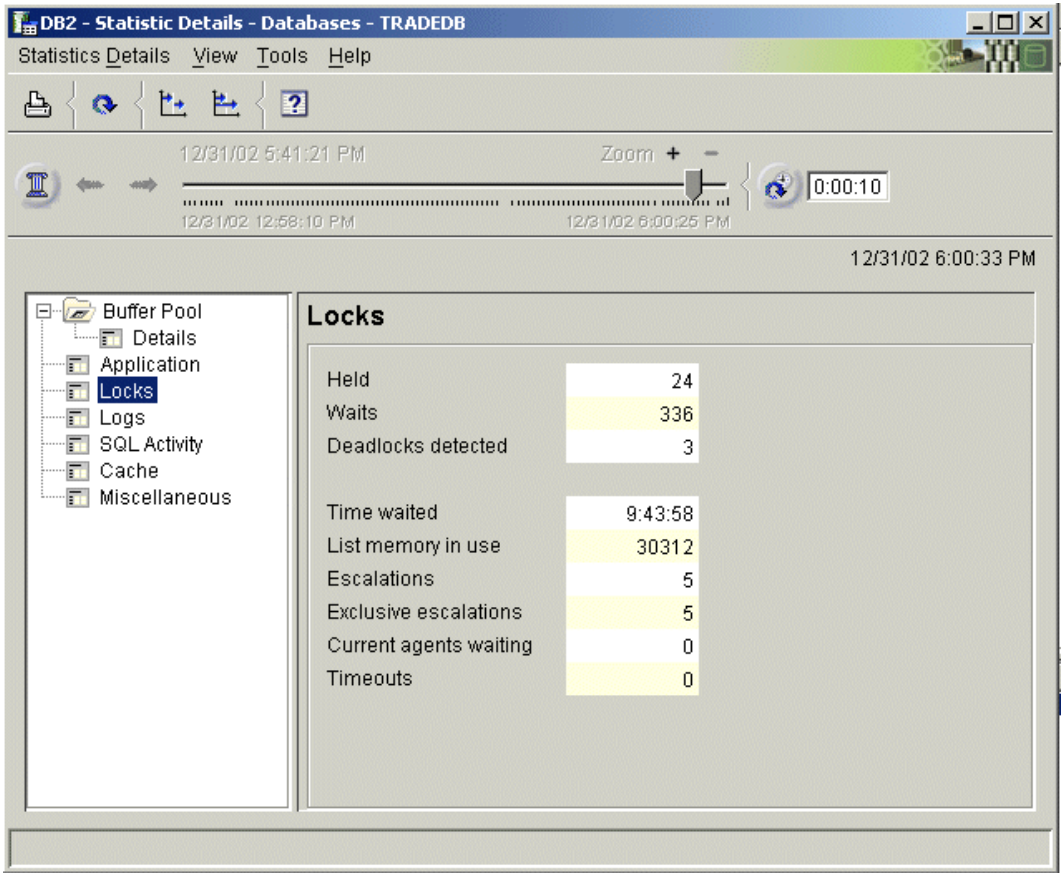


Figure 2-12 Statistics Details of Database Locks

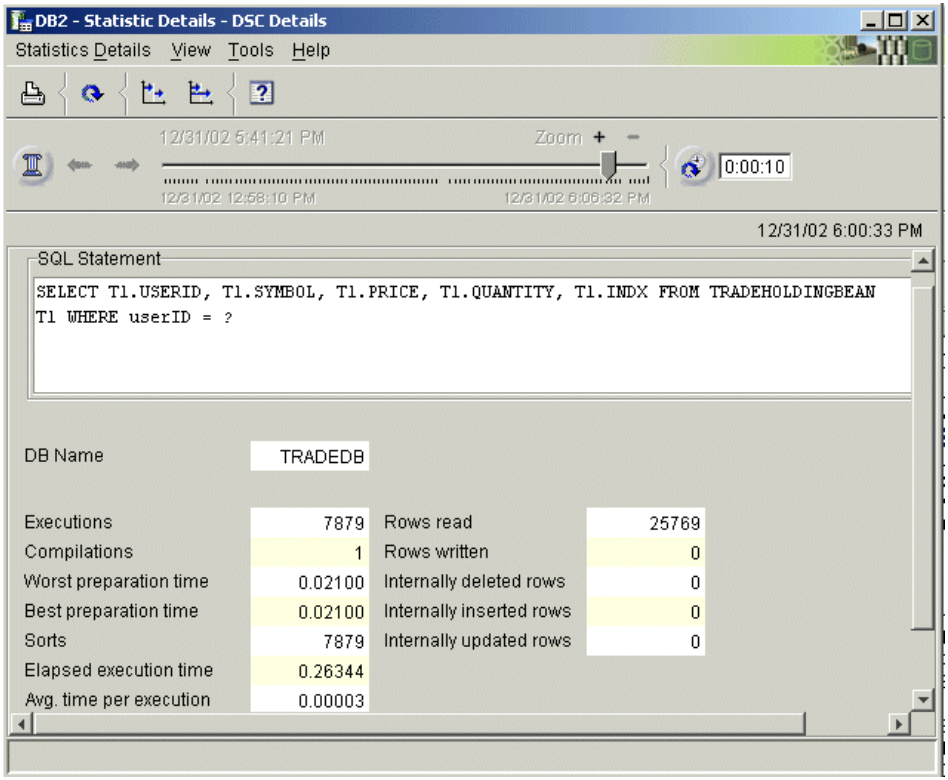


Figure 2-13 Statistic Details - Dynamic SQL Cache Details

System Health

The System Health monitoring function enables statistics details in graphical format. Here too, this information is collected on demand or at the auto refresh interval (default of 6 seconds) when DB2 PE issues a **`db2 get snapshot`** command. However, in order to display data in graphical format, the System Health monitoring function stores “short-term” history of snapshot data in memory.

The history slider can be activated to view System Health at a previous point in time when problems were reported.

The System Health monitoring function can be invoked by selecting the DB2 instance of interest, and then double-clicking **System Health**.

In order to view statistics data in graphical format, you need to create a new Data View, or use one of the predefined Data Views and organize them in one or more Data Groups.

You can create a new Data View in an existing Data Group as follows:

1. In the tree, click **Data Views** and then click **Selected -> New.**
2. Select the **Category** tab to provide a name for the new Data View, and to specify the counter category to be represented by the Data View.
3. Select the **Counters** tab to select the counters or data to be displayed, and to define how this is to be displayed.
4. Select the **Thresholds** tab to define the thresholds for the selected counters.
5. Select the **Graphics** tab to specify the details of the graphical layout.
6. Click **Finish** to save the settings and return to the main window.

Figure 2-14 shows an example of a couple of Data Views.

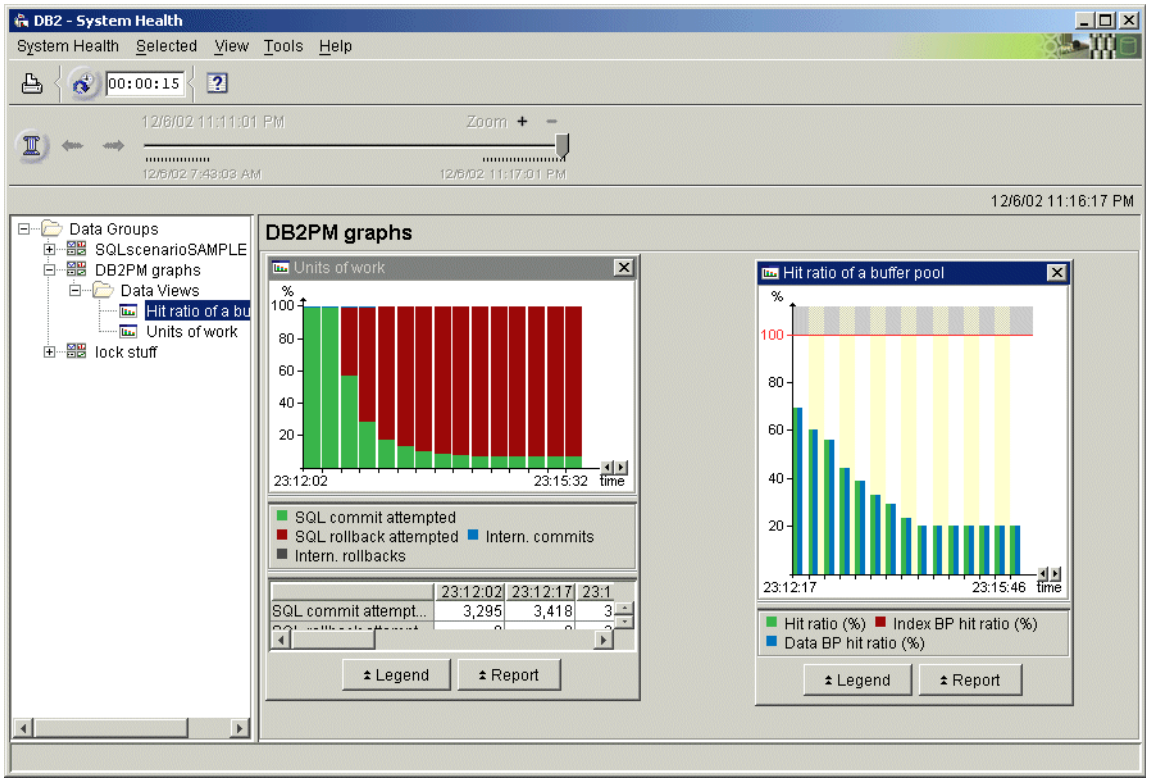


Figure 2-14 System Health data views

Applications in Lock Conflicts

The Application in Lock Conflicts monitoring function displays all the applications involved in lock conflicts. When several threads or applications try to simultaneously access the same data, DB2 uses locks to ensure that data integrity is not compromised. This monitoring function provides lock-related information such as the mode of lock, lock type, and the name of the table that was locked by the application. Here too, this information is collected on demand or at the auto refresh interval (default of 6 seconds) when DB2 PE issues a **`db2 get snapshot`** command.

The history slider can be activated to view Applications in Lock Conflicts at a previous point in time when problems were reported.

The Applications in Lock Conflicts monitoring function can be invoked by selecting the DB2 instance of interest, then double-clicking **Applications in Lock Conflicts**, which displays the window shown in Figure 2-15.

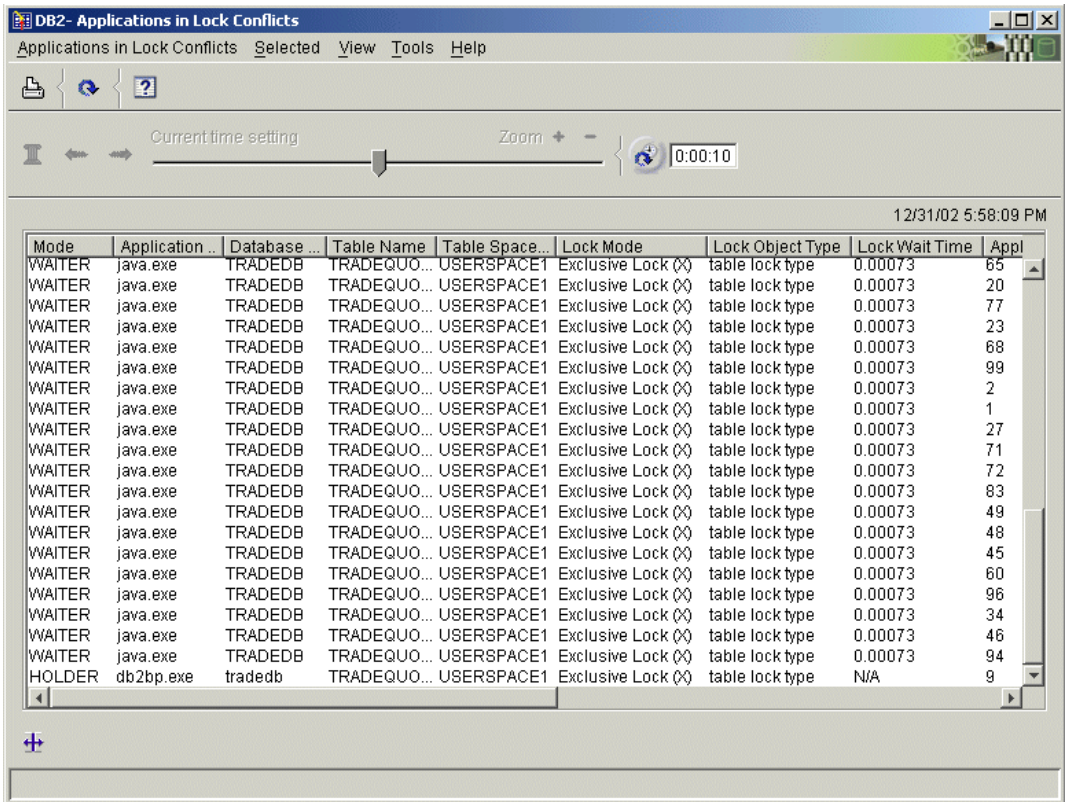


Figure 2-15 Applications in Lock Conflicts

Locking Conflicts

The Locking Conflicts monitoring function displays all the applications associated with a specific resource. Here too, this information is collected on demand or at the auto refresh interval (default of 6 seconds) when DB2 PE issues a **`db2 get snapshot`** command.

The history slider can be activated to view Locking Conflicts at a previous point in time when problems were reported.

The Locking Conflicts monitoring function can be invoked by selecting the DB2 instance of interest, then double-clicking **Locking Conflicts**, which displays the window shown in Figure 2-16**.**

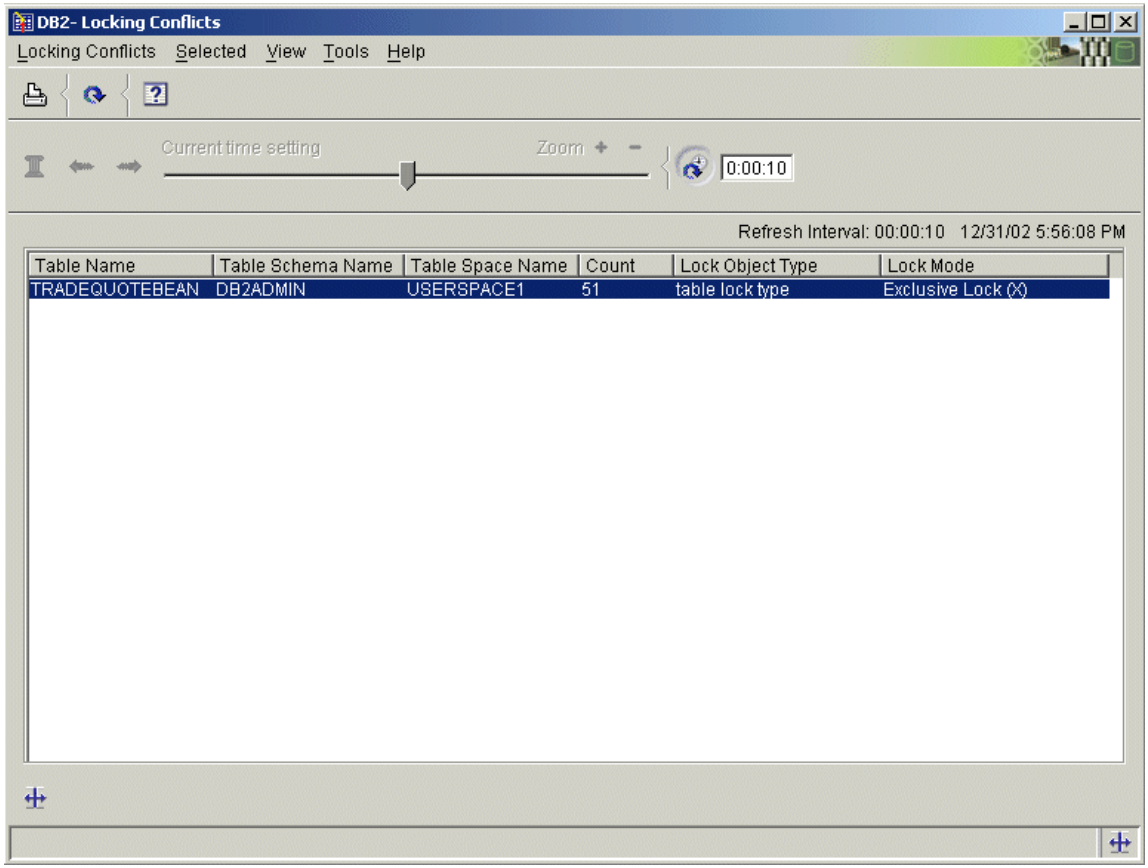


Figure 2-16 Tables in Locking Conflicts and the Locking Mode

System Parameters

The System Parameters monitoring function provides details of the DB2 instance configuration parameters and database configuration parameters. This information is collected on demand or at the auto refresh interval (default of 6 seconds) when DB2 PE issues a **`db2 get snapshot`** command.

The history slider can be activated to view System Parameters information at a previous point in time when problems were reported.

The System Parameters monitoring function can be invoked by selecting the DB2 instance of interest, then double-clicking **System Parameters**, which displays the current DB2 instance parameters window shown in Figure 2-17.

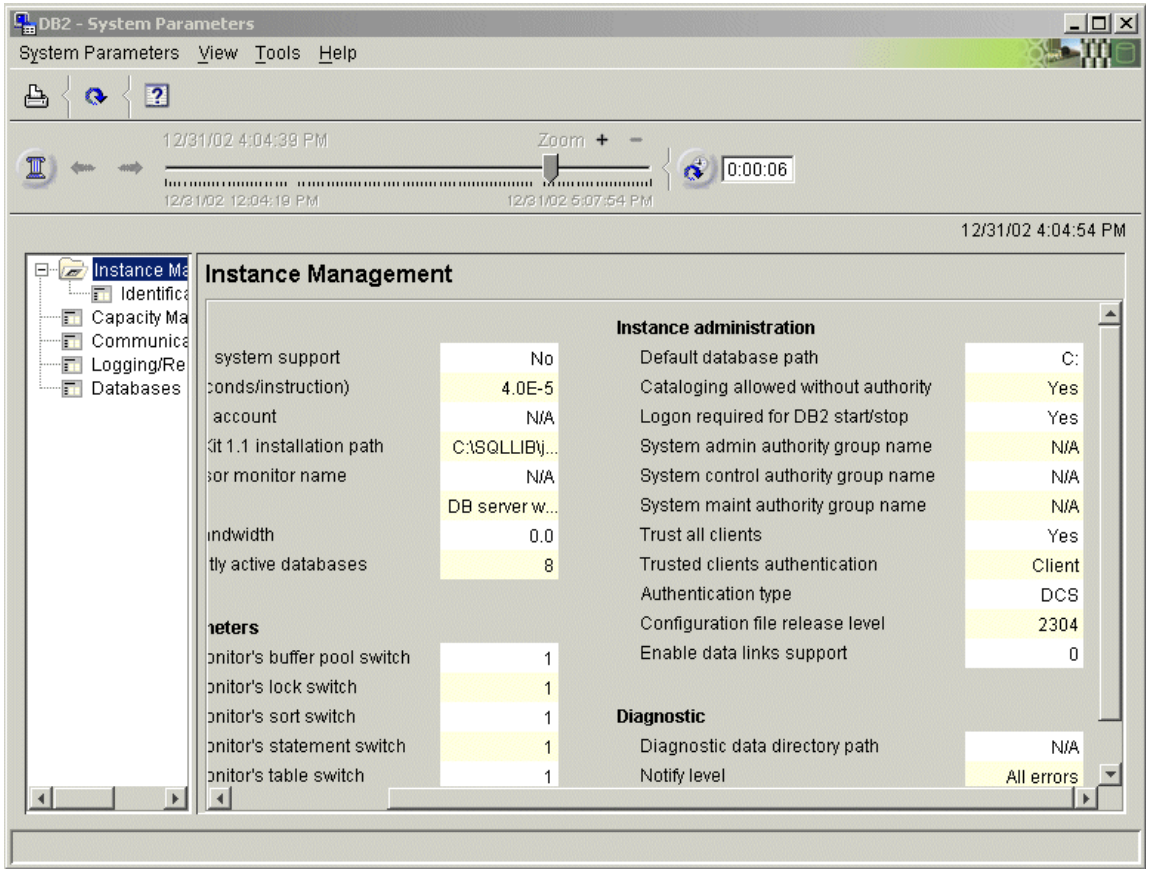


Figure 2-17 System Parameters at DB2 instance level

The window shown in Figure 2-18 shows the System Parameters at the database level. You can view this by double-clicking the database in System Parameters at Instance Level.

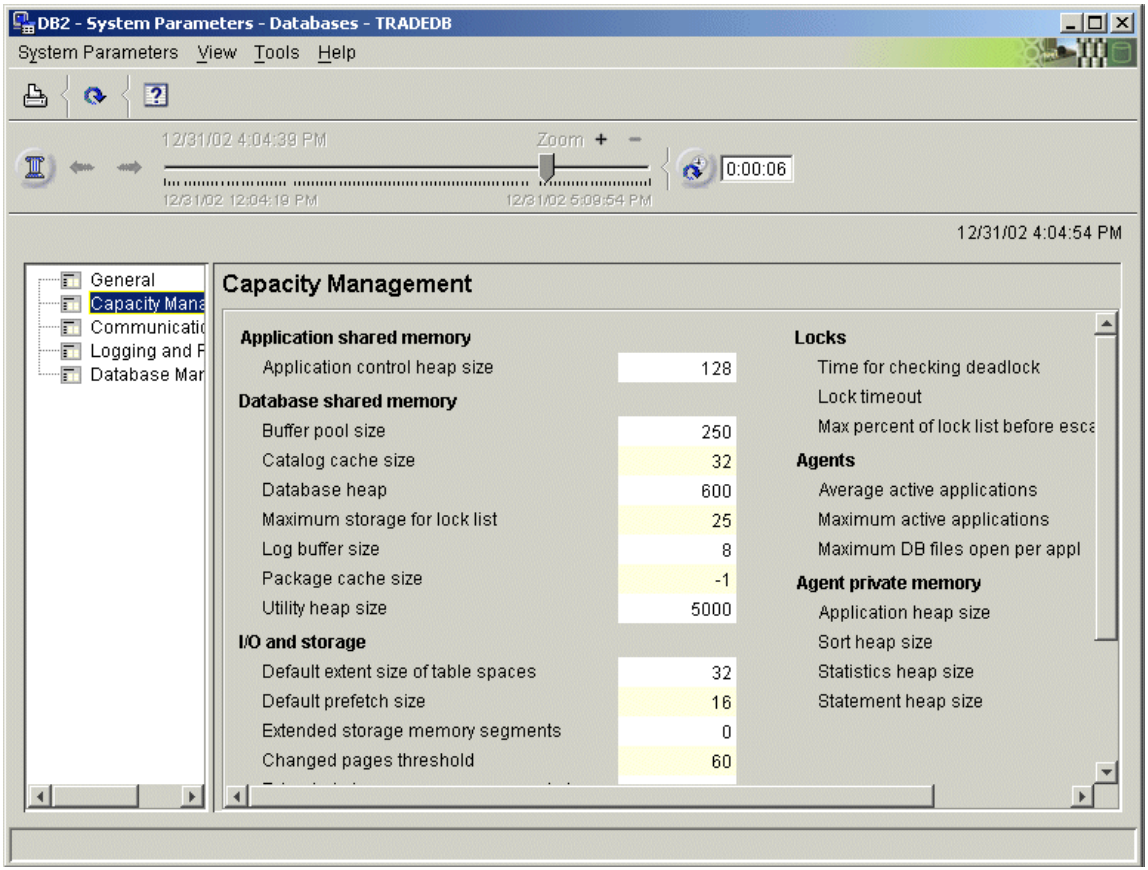


Figure 2-18 System Parameters at the database level

Performance Warehouse

Performance Warehouse provides a process-oriented view of performance analysis tasks. It allows you to automate tasks that previously required user interaction, such as loading DB2 data into a performance database or generating reports.

Performance Warehouse can be used to generate, store and analyze DB2 performance data. You can define, schedule and run processes that automate the generation of performance data, loading of this data into a performance warehouse database, and create reports against this data in the performance

warehouse. You can also use SQL queries to analyze and report performance data stored in the performance warehouse.

DB2 PE provides sample processes and queries to help you get started.

Important: DB2 PE creates and deletes DB2 event monitors to collect the data that is stored in the performance warehouse database. Every initiation of a collection process in the Performance Warehouse monitoring function involves event monitor creation and deletion. Therefore, performance data collection by this monitoring function should be used with care to avoid undue performance overhead on the monitored system.

We briefly describe the steps involved in invoking the Performance Warehouse monitoring function, and generating, loading and reporting performance data.

Invoking Performance Warehouse

The Performance Warehouse monitoring function can be invoked by selecting the DB2 instance of interest and then double-clicking **Performance Warehouse.**

It may also be invoked from the System Overview window by clicking **Tools-> Performance Warehouse**.

The Performance Warehouse main window is shown in Figure 2-19 on page 37.

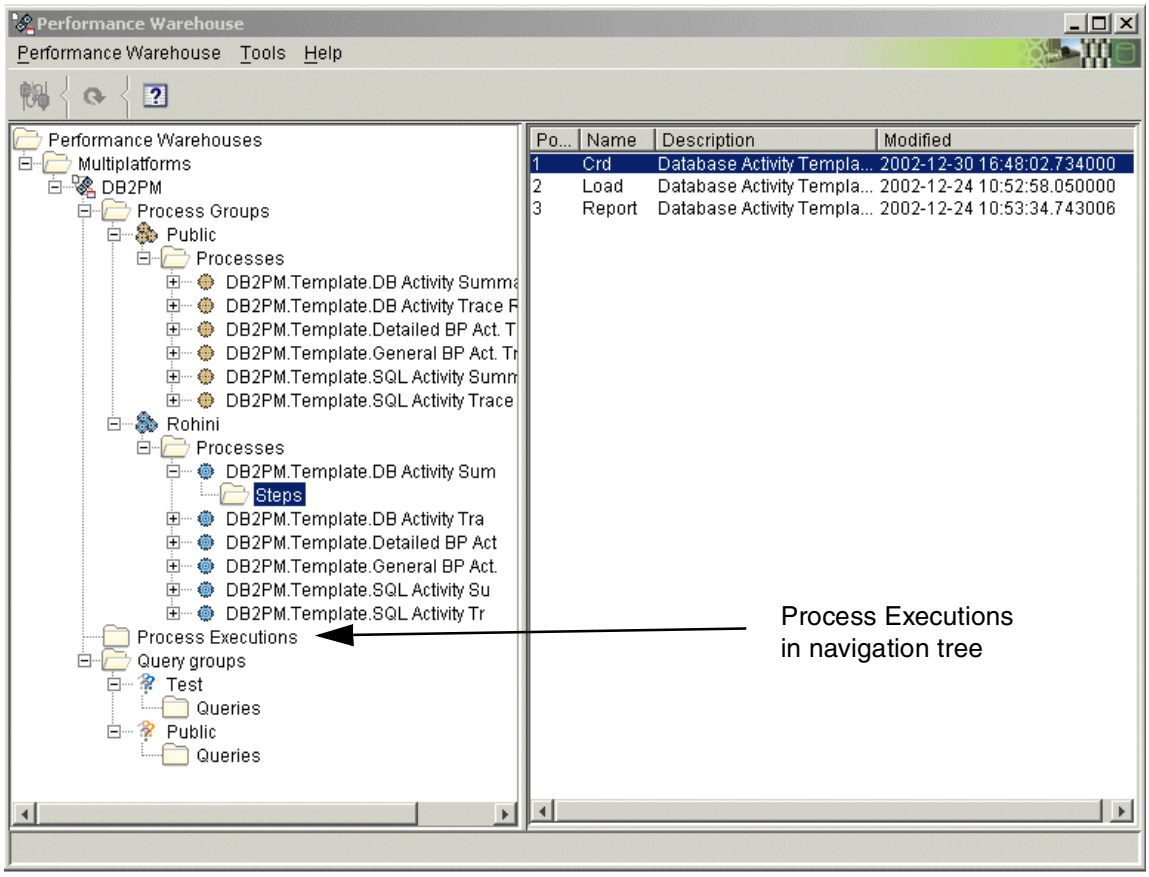


Figure 2-19 Performance Warehouse main window

The following steps are involved in creating a process for generating performance reports using the Performance Warehouse monitoring function.

1. Create a process group
2. Define the process
3. View/modify the process properties
4. Configure the process steps
5. Run the process
6. View the output report
7. Define and run queries

Each of these steps is described in the following sections.

Create a process group

A process needs a process group as a container. To create a process group:

1. In the tree of the Performance Warehouse window, click **Process Groups** and then click **Selected -> Create.**
2. In the Name field, type a unique name for the process group.
3. The *Author, Creation date and time* and *Modification date and time* fields will be filled in after you click **OK**. The Author field will contain your DB2 User ID.
4. Click **OK** to save the process group and return to the Performance Warehouse main window. The new group appears in the tree as a sub-item of Process Groups.

Define the process

Process templates are supplied with the DB2 PE. These templates are located in the Public process group. These templates should be used to define your required process. To create a process:

1. In the tree of the Performance Ware house window, under Public, click the template you require and then click **Selected -> Copy.**
2. In the Copy Process window, select the process group for *New Process group* and type a unique name for the copied process for *New process name*.
3. Click **OK**.

When you return to the Performance Warehouse window, the template's steps are also copied to the new process.

View/modify the process properties

To view the properties of the newly created process:

1. In the tree of the Performance Warehouse window, click the newly defined process, and then click **Selected -> Properties.**
2. Check that the following are set:
   - **Status** is set to `in definition` so that you can make changes to the process.
   - **Scope** is set to `private`, so that this process is not available to other users.
   - **Schedule** of the process is set to `immediately`, since we intend to initiate this process as soon as we have completed it creation. DB2 PE does provide the option to schedule this process at user-defined intervals (this is not discussed here).
3. Click **OK** to save the changes to the process and return to the Performance Warehouse window.

Configure the process steps

Processes consist of one or more steps that perform tasks, such as collecting performance data, loading it into the performance warehouse database, and generating reports from the loaded data. These steps are covered here.

1. The *Collect Report Data step* reads an existing save data set and produces a save-file data set that can be used as input for a load step. This Collect Report Data Step Property window is shown in Figure 2-20. To configure the CRD step:
   a. In the tree view, click **Steps**. In the right of the window, click the CRD step and then click **Selected -> Properties.**
   b. Click the **Options** tab.
   c. In the Database Name field, specify the name of the database to be monitored.
   d. In the Event monitor name field, type a unique name for the event monitor
   e. In the Event monitor file size in MB field, specify the maximum size for the file to hold the data. When this size is exceeded, the event monitor stops collecting data
   f. In the Elapsed time field, type the time range during which data is to be collected.
   g. For a Buffer Pool Activity or Database Activity Report, provide a value for the Flush interval. This interval specifies the frequency at which in-memory data collection is written out as a record to a file.

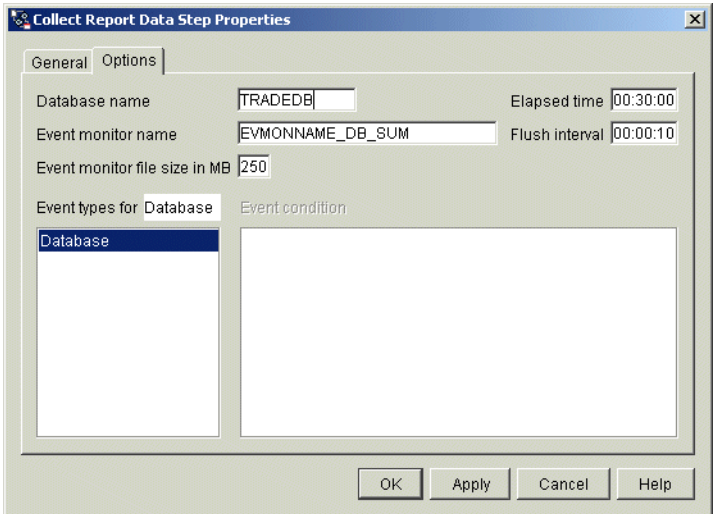


Figure 2-20 Performance Warehouse - STEP1 - Collect Report Data

2. The *Load step* loads an existing file or save-file data set into the performance warehouse database.
3. The *Report step* generates the appropriate reports from the collected data. The Report Step Property window is shown in Figure 2-21. To define the filter in the Report step:
   a. Click the Report step and then click **Selected -> Properties.**
   b. Select the **Options** tab
   c. Select the button next to Filter.
   d. Use the Report Filter window to reduce the data shown in the individual report sections.
   e. Click **OK** to save the changes.

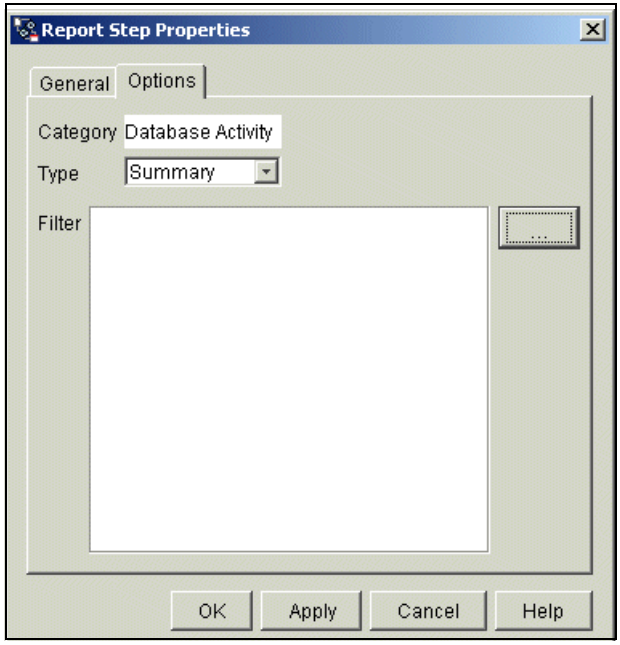


Figure 2-21 Performance Warehouse - STEP3 - Report Properties

Run the process

The process definition is now complete, and the process can now be initiated.

To run the process immediately:

1. Select the process in the tree view, and then click **Selected -> Execute.** The process is scheduled to start immediately and it is activated.
2. In the tree view, click **Process Executions.** The Process Executions page is displayed as shown in Figure 2-22.

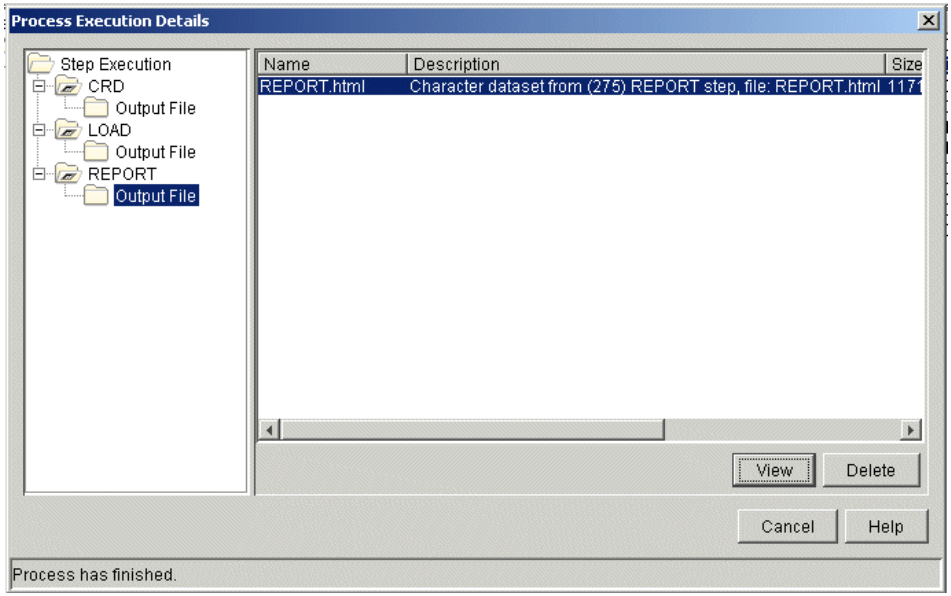


Figure 2-22 Performance Warehouse - Process Execution

View the output report

When the process status changes to `FINISHED` on the Process Executions selection of the navigation tree, as shown in Figure 2-19 on page 37, the output can be viewed in the default browser, as shown in Figure 2-23.

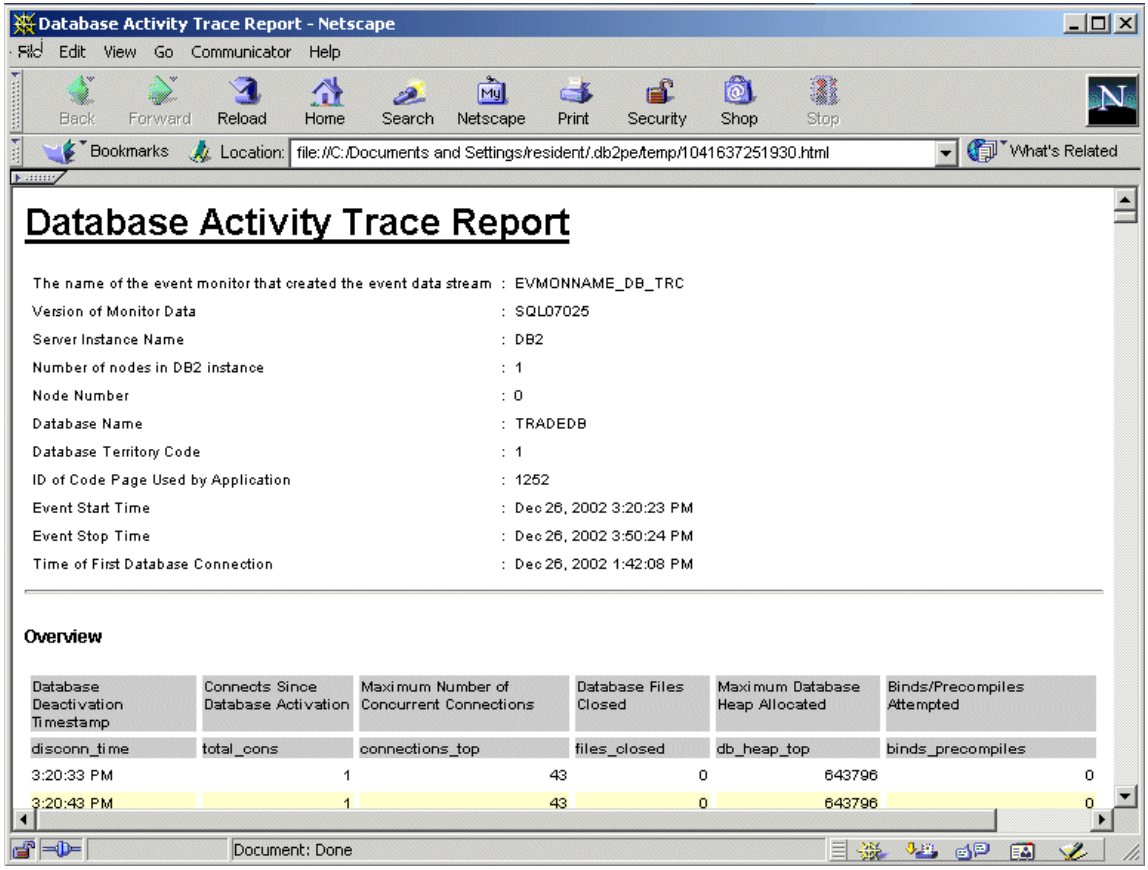


Figure 2-23 Report generated by Performance Warehouse

Define and run queries

A query group is a container for queries. The Public query group contains the sample queries delivered with DB2 PE.

To define a query group:

1. In the tree of the Performance Warehouse window, click **Query Groups** and then click **Selected -> Create**
2. In the Name field, type a unique name for the query group.

3. The *Author*, *Creation date and time*, and *Modification date and time* fields will be filled in after you click **OK**. The *Author* field will contain your DB2 user ID.
4. Click **OK** to save the process group and return to the Performance Warehouse window.
   The new query group appears in the tree as a sub-item of Query Groups.

You can create a new query or you can copy the existing queries from the Public Query Group.

To run the query:

1. In the tree of the Performance Warehouse window, click the appropriate query group and then click its sub-item **Queries**.
2. Click the query you want to run and then click **Selected -> Execute**.
   The Query Execution window appears as shown in Figure 2-24.

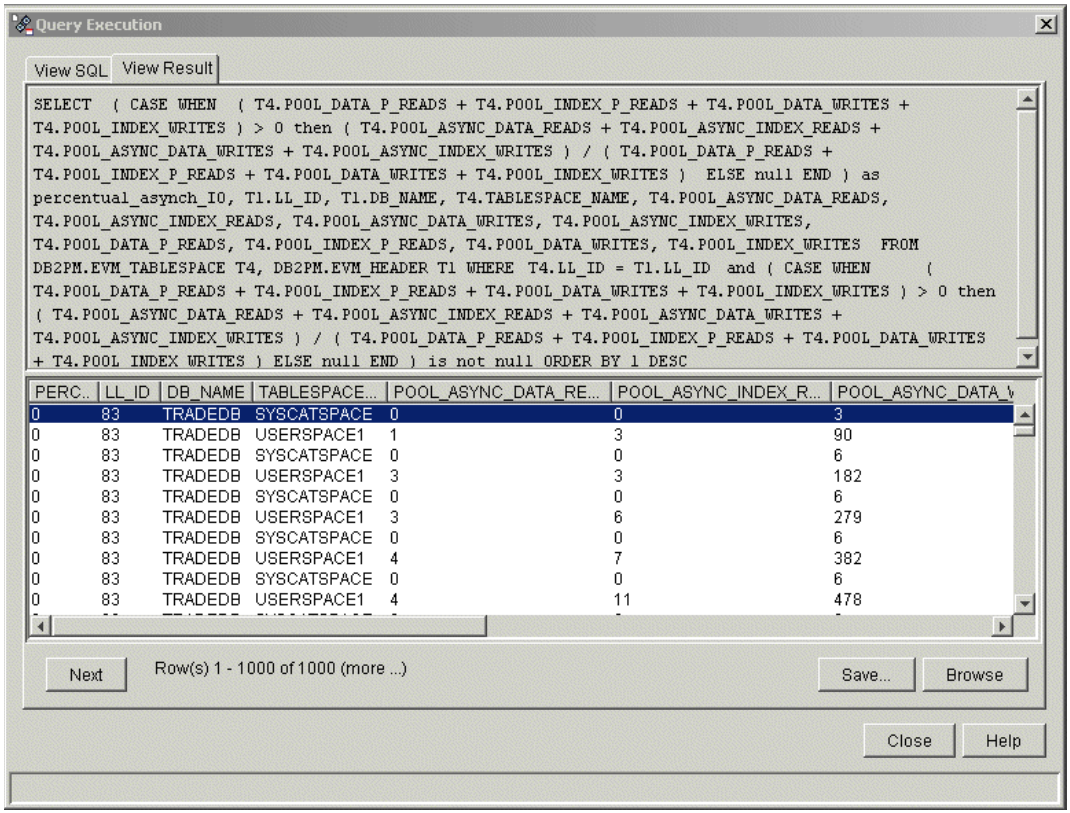


Figure 2-24 Performance Execution - Query Execution

The View SQL tab shows the query definition, which can be edited as required, such as with the filter criteria and any unresolved variables in the query expression.

After the query has run, the Query Execution window View Result page is displayed. The query result can then be saved or viewed in a Web browser.

Buffer Pool Analysis

Buffer Pool Analysis helps you manage buffer pools more efficiently by providing detailed information about their usage.

The Buffer Pool Analysis monitoring function can be invoked by selecting the DB2 instance of interest and double-clicking **Buffer Pool Analysis** to view the main window shown in Figure 2-25.

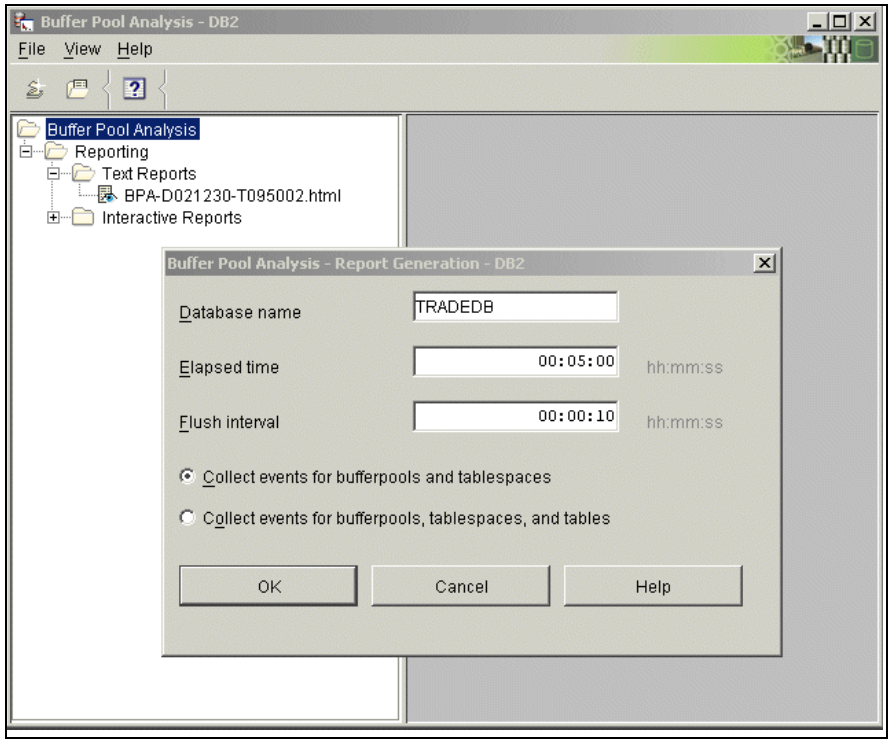


Figure 2-25 Buffer Pool Analysis - Report Generation Settings

Buffer Pool Analysis lets you report and analyze the performance data of a DB2 system.

Note: Like Performance Warehouse, this monitoring function also creates and deletes DB2 event monitors, and the same considerations apply.

To generate a Buffer Pool Analysis report:

1. Select **File -> Generate new Report**.
2. Enter the database name for which the report needs to be generated.
3. Specify the Elapsed Time and Flush Interval to collect the data.
4. Click **OK** to save the information and return to main window.

The Report Generation window appears as shown in Figure 2-26.

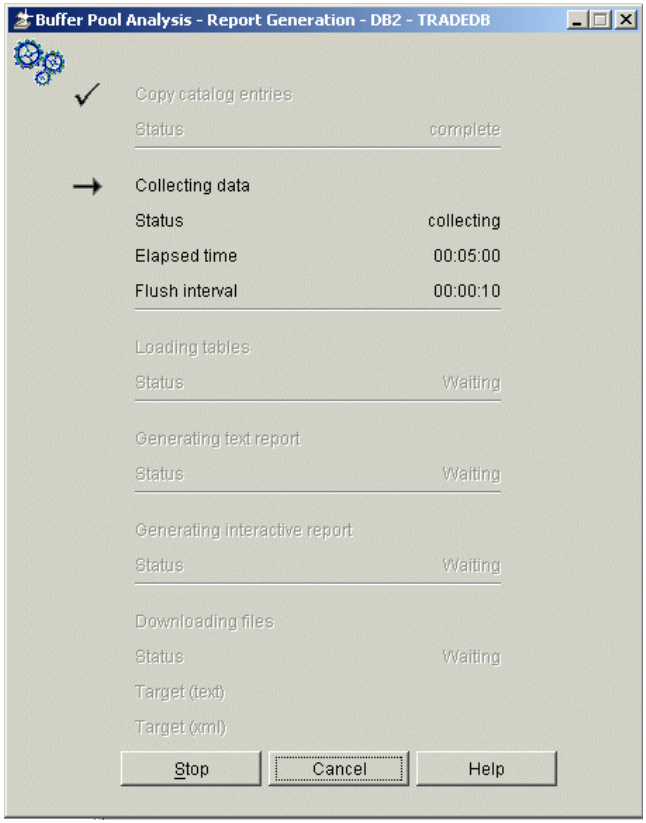


Figure 2-26 Buffer Pool Analysis - Report Generation

At the completion of this process, select the appropriate execution report in Interactive Reports in the navigation tree, and highlight **Summary** to view the report shown in Figure 2-27.

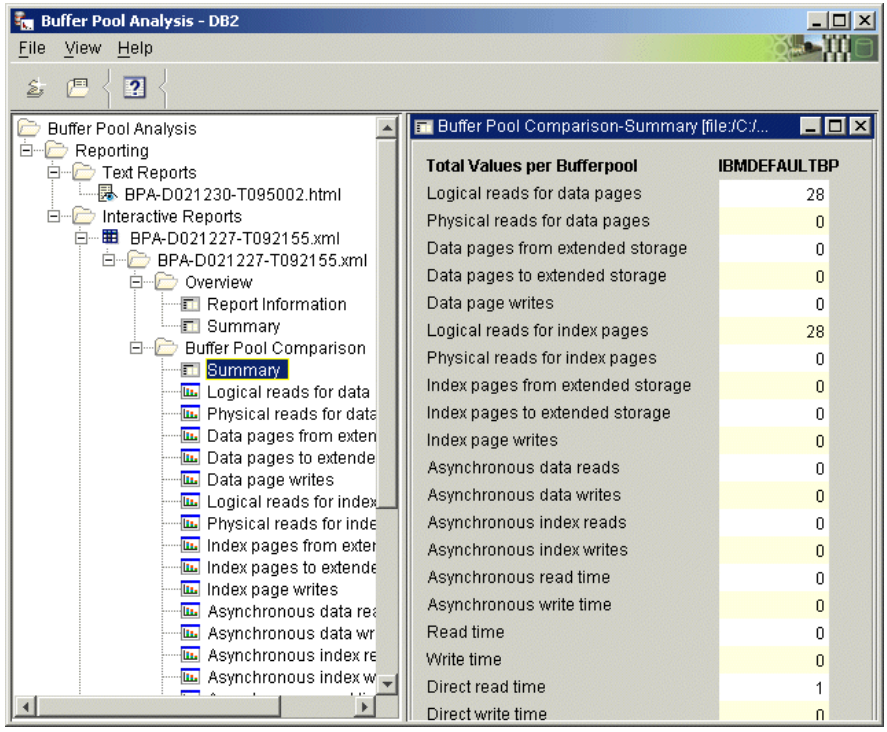


Figure 2-27 Buffer Pool Analysis - Summary of Sample Report

To view the graphical form of the counters, in the tree, double-click the counter, for example, **Logical reads for data pages**. Figure 2-28 shows an example of the graphical representation.

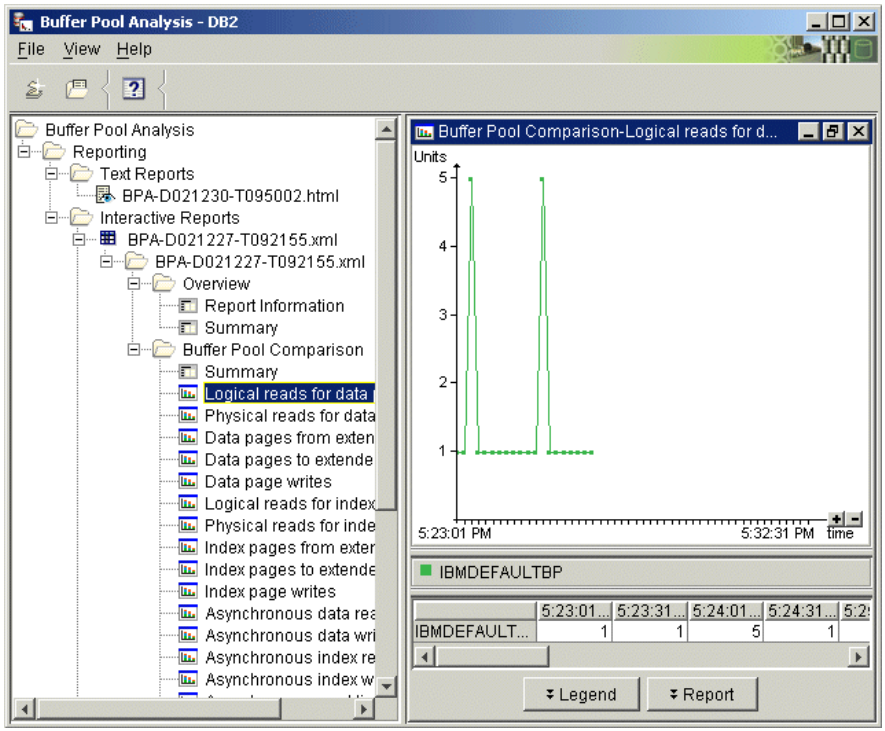


Figure 2-28 Buffer Pool Analysis - Graphical Report

2.3.3 Data flow in DB2 PE

Figure 2-29 describes the flow of data through DB2 PE and the various persistent and transient caches where monitored data is stored.

- History mode snapshot data that is collected at the DB2 instance level is stored in two history files that are used in wraparound mode. This data is semi-persistent since it can be overwritten when the history files' wraparound occurs. The contents of these history files are viewed in the DB2 PE Client using the history slider.
- System Health "short term" snapshot data is stored in memory on the DB2 PE Client machine and is therefore transient. This means that System Health data views' short term history is lost when the DB2 PE Client is restarted.
- Event Monitor data that is generated by the Performance Warehouse and Buffer Pool Analysis monitoring functions is stored in Performance Warehouse tables on the DB2 server. This data is persistent.
- The various processes, diagnostic traces, DB2 PE Client setup, queries and reports are stored in a folder on the DB2 PE Client, and is therefore persistent. The default directory on Windows is for this information is C:\Documents and Settings\<userid>\.db2pe\

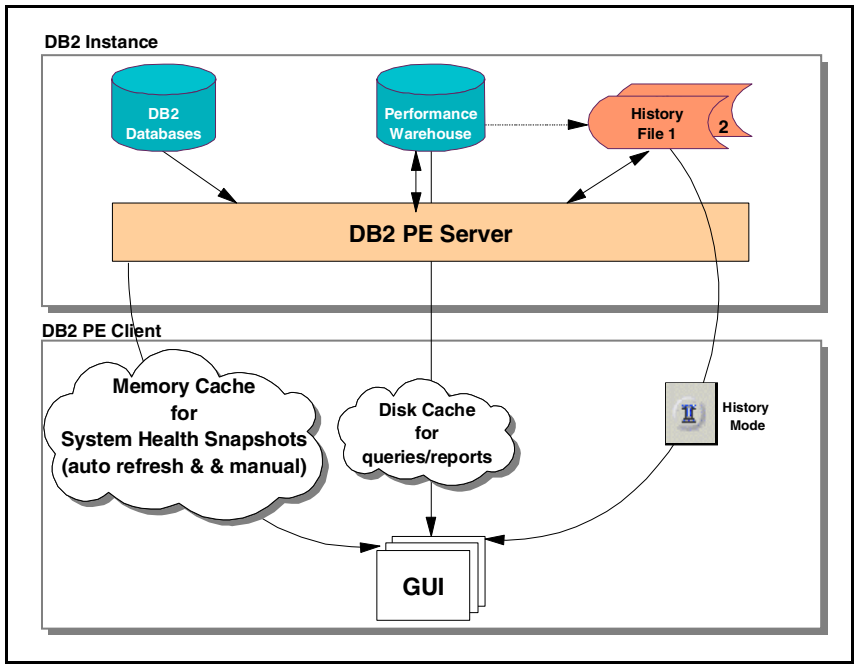


Figure 2-29 Data flow through DB2 Performance Expert

2.3.4 DB2 PE menu items vis-a-vis types of monitoring

Table 2-1 describes the relationship between the various DB2 PE features and functions, and the routine, online/realtime event, and exception monitoring strategies adopted by DBAs in their performance management function.

As mentioned earlier, performance data collected by DB2 PE is either a result of a **`get snapshot`** command, or the creation of an "event monitor". Typically, **`get snapshot`** incurs less overhead than an "event monitor". However, the actual degree of overhead associated with each approach depends upon the type of event monitor requested, the application workload, and the frequency and duration of execution of such requests, and therefore such information collection should be kept to a minimum without sacrificing valuable information gathering.

In Table 2-1, the issuing of the **`get snapshot`** command occurs on demand, or at a user-defined interval that can be different for each of the available tasks.

Table 2-1 DB2 PE menu items vis-a-vis monitoring strategies

Menu Items in DB2 PE System Overview	Exception monitoring	Online monitoring	Routine monitoring
Application Summary (Result of **`get snapshot`** command)		Displays application level detail such as CPU time, elapsed time, buffer pool hit ratio, and SQL activity	
Statistics Details (Result of **`get snapshot`** command)	Displays system wide statistics information such as instance, database, tablespace and buffer pool activity		
System Health (Result of **`get snapshot`** command)		Displays graphs of counters such as hit ratios, heap utilization, and SQL activity	
Application in Lock Conflicts (Result of **`get snapshot`** command)	Displays all applications involved in lock conflicts		

Menu Items in DB2 PE System Overview	Exception monitoring	Online monitoring	Routine monitoring
Locking Conflicts (Result of **`get snapshot`** command)	Displays detailed table level locking conflicts		
System parameters (Result of **`get snapshot`** command)	Displays database manager configuration and database configuration parameters		
Performance Warehouse (Creates an “event monitor” to collect data)	Collects trace data about database, buffer pool and SQL activity		Use in short bursts only
Buffer Pool Analysis (Creates an “event monitor” to collect data)	Collects trace data about buffer pool activity		Use in short bursts only

3

DB2 Performance Expert configuration considerations

In this chapter, we describe the key considerations involved in configuring DB2 Performance Expert in your environment and provide recommendations to achieve optimal use of the product.

The topics covered are:

- DB2 PE architecture review
- DB2 PE Client considerations
- DB2 PE Server considerations

3.1 DB2 PE architecture review

Understanding the architecture of DB2 PE vis-a-vis the monitored DB2 environment and the collection, storage and flow of information between the various components of DB2 PE will help you determine the configuration best suited to your environment.

Figure 3-1 describes the general environment structure of DB2 PE for Multiplatforms.

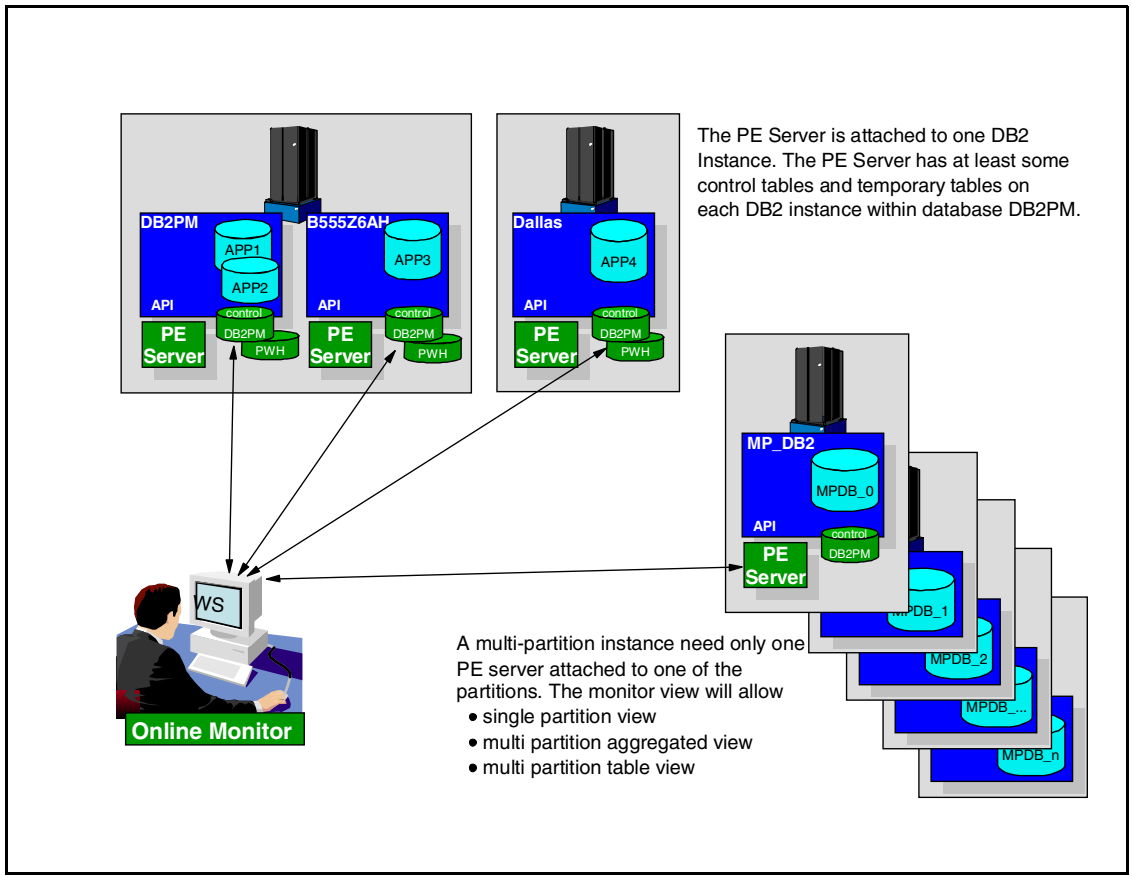


Figure 3-1 DB2 PE for Multiplatforms environment structure

The key points to be noted here are as follows:

- A single DB2 PE Client can monitor one or more DB2 instances running on a heterogeneous mix of Windows and UNIX server platforms. In addition, the target DB2 instances may be a combination of single and multipartition DB2s, as well as a mix of DB2 V7 and DB2 V8 instances.
- One DB2 PE Server is associated with each monitored DB2 instance, and a DB2PM database is associated with this instance.

The following factors must be taken into account when choosing and configuring a DB2 PE environment in your organization:

1. The number of DB2s managed from a single DB2 PE Client, and the overhead generated by it on monitored DB2 instances.
2. Coexistence considerations of the DB2 PE Server code, the DB2PM database and the DB2 instance being monitored. This includes the impact of all these elements on server cycles, disk space consumption, database manager configuration parameters, and database configuration parameters of the databases being monitored as well as DB2PM.

These considerations are discussed in further detail in the following sections.

Note: We did not experiment with DB2 PE's Performance Agent feature, or HACMP environments at all, and only very briefly with DB2 UDB V7 EEE and DB2 UDB ESE V8 multipartitions. Therefore, considerations relating to these topics are *not* included in this redbook.

3.2 DB2 PE Client considerations

Figure 2-29 on page 48 shows that the DB2 PE Client uses memory caches to store System Health short term history. DB2 PE Client is a Java application; therefore, depending upon the number of DB2 PE monitoring function windows open and the desired number of snapshots in System Health, memory consumption on the DB2 PE Client can grow rapidly and negatively impact performance. The management of multiple DB2 instances from a single DB2 PE Client can further exacerbate the demand for CPU cycles and memory.

We therefore recommend the following for the DB2 PE Client:

1. Isolate the DB2 PE Client from the DB2 PE Server to eliminate any possible adverse impact (memory and cycles) that coexistence with the DB2 PE Server may introduce.
2. Use a dedicated Windows machine for the DB2 PE Client with sufficient memory to accommodate multiple open DB2 PE monitoring function windows

and adequate System Health short term history caching (at least greater than 512 MB).

3. If possible, install only a DB2 Runtime Client on the DB2 PE Client machine to avoid unnecessary resource utilization by a DB2 Server.

> **Important:** There are restrictions related to installing a DB2 PE Client on a DB2 V8 codebase, if access to DB2 V7 servers is involved. Please refer to DB2 V8 installation documentation for a detailed discussion of these restrictions.

4. The default value for the auto refresh interval on the DB2 PE Client is `6` seconds for all the DB2 PE monitoring functions; this can potentially result in significant overhead on the monitored DB2 instance as well as the DB2 PE Client. We therefore suggest that you either deactivate auto refresh (the default for auto refresh is `OFF`) and use manual refresh as required, or increase the auto refresh interval to an acceptable value through trial and error.

> **Important:** When the DB2 PE environment is configured to access a single DB2 instance from multiple DB2 PE Clients, simultaneous setting of certain DB2 instance level parameters may cause certain updates to be lost.
>
> Therefore, when multiple DBAs are allowed to manage different databases in the same DB2 instance, a process should be in place to ensure that only one DBA is designated to update instance level parameters to avoid potential conflicts.

3.3 DB2 PE Server considerations

The DB2 PE Server collects, stores and delivers information based on DB2 PE Server configuration settings and DB2 PE Client requests.

By virtue of the fact that DB2 PE Server coexists with the DB2 instance being monitored, configuration considerations can be divided broadly into the following two categories:

- Monitored DB2 instance/database related
- DB2 PE Server related

3.3.1 Monitored DB2 instance/database related

DB2 PE Server is a Java application accessing the target DB2 instance and associated databases, including the DB2PM database. Depending upon the number of monitoring requests initiated, multiple agents will need to be assigned in the database engine to service these requests. This may require modifying the DB2 instance's DBM configuration parameters as well as the target databases' DB configuration parameters.

Important: We strongly recommend that you review the DB2 instance settings before and after the installation of DB2 PE to ensure that the various parameter settings are correct.

- DB2 instance DBM configuration parameters that may need to be increased include JAVA_HEAP_SZ, MON_HEAP_SZ and MAXAGENTS.
- Target database configuration parameters that may need to be increased include APPLHEAPSZ and MAXAPPLS.
- DB2 PE Server accesses the DB2 engine to get information about performance data. The type and quantity of information returned is controlled by DBM parameters known as *monitor switches*, such as DFT_MON_SORT, DFT_MON_LOCK, DFT_MON_TABLE. DFT_MON_BUFPOOL, DFT_MON_UOW and DFT_MON_STMT. These switches may be set by **db2 update dbm cfg** commands or by modifying the Java property file db2pesrv.prop, which is located in the <installation directory>\instances\<DB2 instance> for Windows, and /var/db2pe//<DB2 instance> for UNIX. The advantage of making changes to the Java property file is that the changes are effective immediately, and you do not have to start and stop the database manager.

  Important: The interplay between the DBM monitor switch settings and the db2persrv.prop monitor switch settings currently varies between DB2 V7 and DB2 V8.

  We urge the reader to consult the latest documentation on this subject in the README files distributed with the fixpaks, and the following Web site:

  `http://www.ibm.com/software/data/db2imstools/support.html.`

  We strongly recommend that you conduct accurate tests with the various switch settings in your environment to ensure that the information you require for performance analysis is collected by DB2 PE.

The incremental value to be applied to these parameters depends upon the nature of the application workload on the monitored system, the nature of the

monitoring expected by the tool, and the slack built into the configuration parameters prior to the introduction of DB2 PE in the target environment.

Note: Given that each organization's workload and environment can vary significantly, we recommend that you use "business as usual" monitoring and tuning techniques to determine appropriate incremental values for these parameters in your specific environment.

3.3.2 DB2 PE Server related

The two main configuration items in this category are history file size, interval and multiplier considerations, and the database configuration parameters for the DB2PM database.

History file

"History mode" on page 21 provides an overview of this functionality.

The interval, multiplier, and size of the history file determine how far back in history one will be able to go in the DB2 PE Client. The default size of each history file (named fpesnap.dat1 and fpesnap.dat2) is 50 MB, thus providing a total of 100 MB for history.

Note: Here again, given that each organization's workload and environment can vary significantly, determining the interval, multiplier, and size of each history file most appropriate for your installation will have to be a trial and error exercise.

The size of the history file can be modified in realtime by changing the value using the GUI. If both files already exist, the next time a snapshot is taken, DB2 PE Server will check the setting of the archive size and adjust the file accordingly. For example, if file fpesnap.dat1 is 50 MB, and fpesnap.dat2 is 20 MB (current active file), and you change the archive setting from 50 MB to 200 MB, then DB2 PE Server will continue to write to the fpesnap.dat2 file until it reaches the 200 MB limit. When it switches over to fpesnap.dat1, it will extend that to 200 MB as well, thus consuming a total of 400 MB.

Shrinking of the archive size in realtime is also possible. For example, if the archive files are currently 200 MB and 175 MB (current active file), and you reduce the size to 150 MB, then DB2 PE Server will recognize that the current file is over the limit, and switch over to the other file, which will be limited to 150 MB. The previous 175 MB file will eventually be reduced to 150 MB when the next wraparound occurs.

These history file parameters may be modified using the DB2 PE Client GUI or by issuing SQL update statements directly against the PARAMETER and HISTORYDATA tables in the DB2PM database. An example of the SQL for updating the PARAMETER table follows:

```
connect to db2pm
update db2pm.parameter set pa_flagvalue = 'Y' where pa_key =
'SNAPSHOTTRACE'
update db2pm.parameter  set pa_intvalue = value  where
pa_key='HISTORYINTERVAL'
update db2pm.parameter set pa_intvalue = value where  pa_key =
'HISTORYSIZE'
```

DB2PM database

The DB2PM database stores control information for DB2 PE Server, as well as event monitor performance data collected using the Performance Warehouse and Buffer Pool Analysis monitoring functions.

DB2 PE Server sets the appropriate DB2PM database configuration parameters during its installation, but our experimentation indicated that you should monitor and consider increasing DB2PM's LOCKLIST, MAXLOCKS, APPLHEAPSZ and MAXAPPLS database configuration parameters.

4

Usage scenarios

In this chapter, we identify problems commonly encountered in a DB2 UDB environment and describe scenarios using DB2 PE to identify and resolve such problems.

This chapter contains the following:

- Introduction
- Exception monitoring scenarios
- Online/realtime event monitoring scenarios
- Routine monitoring considerations

4.1 Introduction

One of the main objectives of an IT organization is to ensure that its infrastructure delivers the required performance so that business objectives are continuously met in a constantly evolving and changing business environment. This requires the IT professional to adopt a strategy that is both *proactive* and *reactive* to conditions and events that would tend to adversely impact IT systems performance, and thereby the business.

The *proactive* effort involves a number of tasks, including the following:

- Capacity planning of IT resources
- Choosing the most effective IT architecture for the current and anticipated workload
- Adopting best practices in application design, development and deployment
- Performing rigorous regression testing prior to deployment in a production environment
- Performing routine monitoring of key performance indicators to forestall potential performance problems, as well as gather information for capacity planning

The *reactive* effort involves having a well-defined methodology for identifying the root cause of a problem, and resolving the problem by applying best practices.

Important: In this book, we focus on problem determination relating to the DB2 UDB environment using the DB2 Performance Expert tool, and assume that there are no bottlenecks involving hardware, operating system and network resources. In the real world, however, potential shortfalls in these resources must be identified and addressed as well.

In the following sections, we describe commonly encountered problems that have been identified through:

- Exception events such as user complaints about poor response times, login problems, or aborted transactions
- Online/realtime events such as high lock waits/escalations and sort heap limits being exceeded

We have determined that the commonly encountered problems in a DB2 UDB UDB environment include:

- Constrained resources such as CPU, I/O, memory and network bandwidth.
- Locking contention.

- Database shared memory shortages including in buffer pools, sort heaps, application global memory, and catalog and package caches.
- Lack of appropriate indexes and inadequate housekeeping involving activities such as timely execution of runstats and reorgs.
- Poorly written applications such as inefficient database design and inefficient SQL code.

We have included locking contention, sort heap shortages, and lack of appropriate indexes as the problems identified and resolved in our scenarios.

Note: We used the Trade 2 (also known as WebSphere eBusiness Benchmark) and our own SQL SELECT and UPDATE statements in our scenarios; these applications are described in Appendix A, "Sample applications" on page 107.

Important: Users run with routine monitoring levels during normal operations, and only perform exception monitoring involving more detailed traces for short bursts of time to assist with problem diagnosis. This is due to the negative performance impact on applications during detailed tracing.

In our controlled problem determination scenarios, we chose to run with high levels of diagnostic tracing to perform our problem determination. We recognize that this adversely impacts system and application performance, and may color conclusions about relative merits or benefits of the absolute metrics measured; however, our focus was primarily on problem determination and *not* on performance measurements.

Table 2-1 on page 49 describes the relationship between the various DB2 PE features and functions and the different monitoring strategies. Refer to 2.3.4, "DB2 PE menu items vis-a-vis types of monitoring" on page 49 for a brief discussion of the applicability of DB2 PE's monitoring functions to routine, online/realtime events and exception monitoring strategies.

4.2 Exception monitoring scenarios

When faced with exception events such as user complaints about poor performance, following a standard problem determination methodology will ensure predictable and repeatable resolution scenarios.

Figure 4-1 on page 62 describes a typical sequence of steps to be followed when diagnosing performance problems.

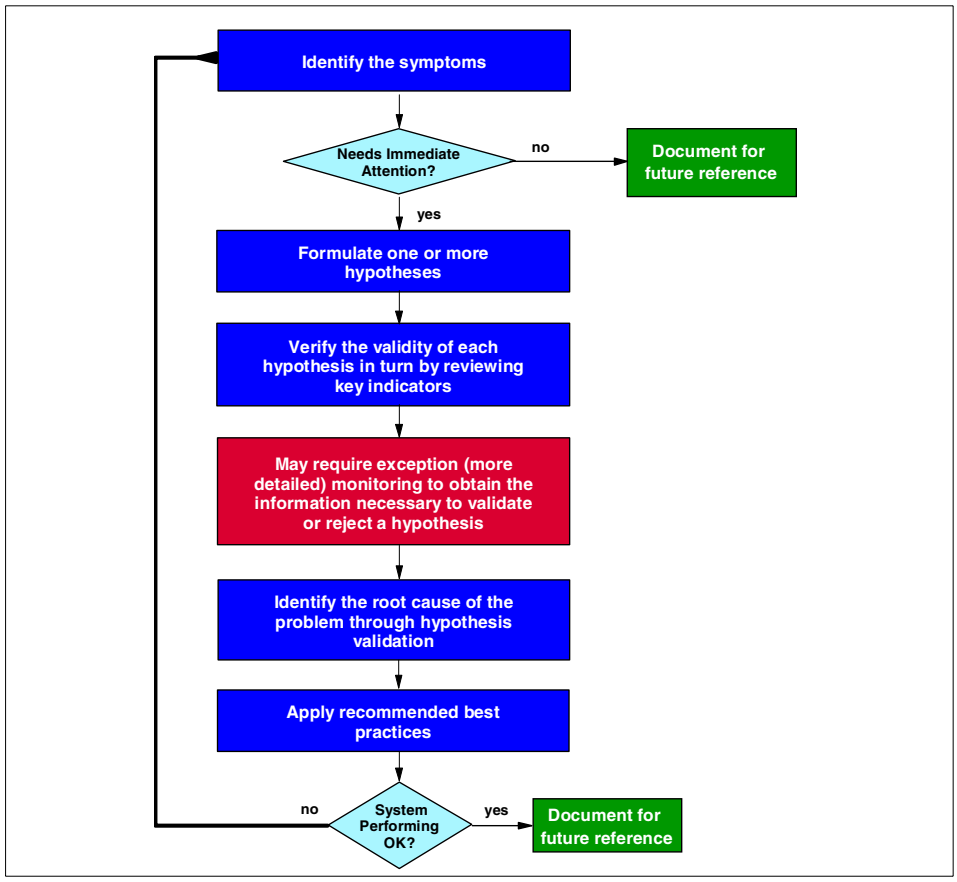


Figure 4-1 A typical problem determination methodology

The entire sequence of steps is triggered by an exception event such as a user complaining about poor response times, or error messages appearing on their screen.

These symptoms must be evaluated for criticality as shown by the decision box ("Needs immediate attention") in Figure 4-1.

- Symptoms that are sporadic and non-disruptive need no immediate action, other than to potentially trigger routine monitoring for possible corrective action. Routine monitoring is covered in this book.
- Symptoms that recur frequently and disrupt business processes require prompt attention to avoid an adverse business impact. We cover some of these scenarios in this book.

- Catastrophic events such as a failure of the system, application server or database server also require immediate attention such as an immediate restart. These events are *not* discussed in this book.

Based on the symptoms and a knowledge base of prior experiences (both external and internal), one should formulate one or more hypotheses as the potential root cause of the problem.

Each hypothesis should then be tested in turn using all available metrics associated with the application under consideration; this includes system resources (such as **nmon** in AIX and Task Manager on Windows), network resources, application server resources, and database server resources. Sometimes, the metrics available from routine monitoring and online/realtime event monitoring may be inadequate to validate or reject a particular hypothesis. In such cases, one may have to request additional diagnostic information through more detailed monitoring levels either on the production system or an available comparable regression system. Such monitoring is often referred to as *exception monitoring*.

Once a hypothesis is validated and the root cause problem has been identified, best practices specific to the root cause problem can be applied to attempt to resolve the problem.

Important: Best practices guidelines are based on user experiences for a given workload and environment, and may or may not provide beneficial results in your particular environment. Therefore, a thorough understanding of the fundamentals of the technical architecture and design is required to explore other alternatives when the documented best practices fail to provide relief. Problem resolution in such cases tends to be an iterative process, where the application of a best practice may result in the manifestation of new symptoms and a formulation of a fresh set of hypotheses.

Once the root cause problem has been resolved, the steps executed and the knowledge gained should become part of the knowledge base to assist in resolving future problem situations.

The next few subsections cover the following problem determination scenarios:

- Lock waits due to the default LOCKTIMEOUT parameter value
- Lock waits and time-outs due to lock escalations (LOCKLIST)
- Long running SQL

We have organized the description of each scenario as follows:

- Description of the application
- Environment configuration
- Monitor level settings
- Workload used
- Triggering event
- Hypotheses and their validation
- Root cause of the problem
- Application of best practices

The following sections describe the setup of the overall environment used in our various problem scenarios. Note that each specific scenario used a subset of the overall configuration.

Description of the application

We used a combination of the Trade2 application (also known as WebSphere Performance Benchmark Sample) and a collection of several “uncommitted” batch jobs with SQL INSERTs, UPDATEs, and SELECTs to cause the problem situations to occur.

A description of Trade2 and SQL used in the batch jobs is given in Appendix A, “Sample applications” on page 107.

Environment configuration

We used a combination of Windows and AIX platforms and DB2 UDB EE V7 and DB2 UDB ESE V8 (single and multipartition) to showcase the capabilities of DB2 PE.

Figure 4-2 on page 65 describes the overall environment.

Two AIX machines were deployed as follows:

- PERSIAN with both DB2 UDB EE V7 and DB2 UDB ESE V8 single partition
- MALMO with DB2 UDB ESE V8, both single partition and multipartition

Four Windows machines were deployed as follows:

- BORON Windows 2000 with the Trade2 application on WebSphere Application Server V4.0.5 and the AKSTRESS tool for driving the application workload. Note that the TRADEDB database was installed on the relevant AIX and Windows boxes for our scenarios involving those platforms.
- NALUR6 Windows 2000 with DB2 UDB V8 Runtime Client for the DB2 PE Client

- NALUR8 Windows 2000 with DB2 UDB V7 EE for both the DB2 PE Server and the DB2 PE Client. Such a configuration would be unusual in a real world environment, and is *not* recommended.
- Laptop Windows 2000 machine with DB2 UDB V8 for the DB2 PE Client

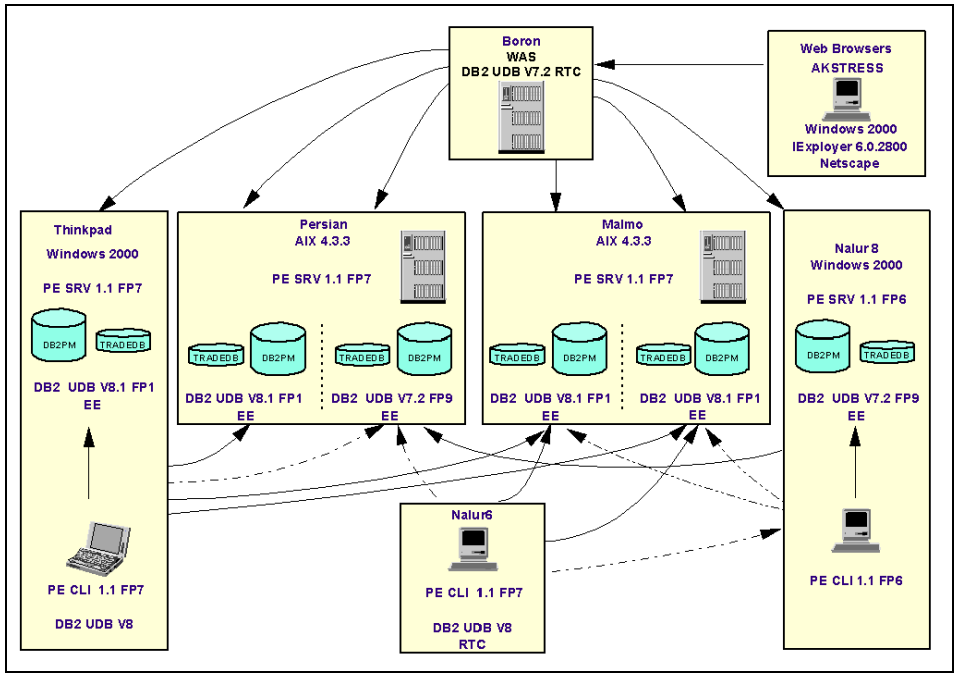


Figure 4-2 Environment setup of Trade2 application

Monitor level settings

We chose default settings for the WebSphere Application Server and TRADEDB database configuration parameter, except for the following:

- DB2 monitor switches
  DB2 PE sets its own monitor switches as required. We chose to leave the defaults in the database manager configuration file for MALMO, and turned them all `ON` on PERSIAN to see what impact they had on the results.
- DB2 DIAGLEVEL
  In real world situations, the default value of `3` would be chosen, and changed to `4` (the highest level of detailed information) only when exception monitoring was required. In our controlled environment, we set this value to `4` in order to simulate exception monitoring conditions which require access to more detailed information for problem diagnosis.

- DB2 PE Interval
  We chose the default value of 60 seconds for the DB2 PE Server snapshot and log to history, but used a category multiplier of 1 in all cases in order to collect all the data with every snapshot.

Workload used

The TRADEDB was populated with more than one million records using the configuration utilities provided with the Trade2 application.

The IBM AKSTRESS tool was used to simulate 200 concurrent user on AIX, and 20 concurrent user on Windows performing login, stock trading and logout tasks.

We wrote batch files with custom SQL statements, and executed them using at least two concurrent command windows.

Triggering event

Typically, the triggers are user complaints about poor response times and application errors.

Hypotheses and their validation

In our controlled environment, we ignored real world root cause possibilities such as network bandwidth concerns, Web server problems, system utilization, and process priorities.

In most cases involving user complaints about performance problems, the typical culprits are:

- Hypothesis 1: Resource constraints
- Hypothesis 2: WebSphere Application Server problems
- Hypothesis 3: Maximum users/connections thresholds exceeded
- Hypothesis 4: Locking problems: waits, deadlocks, timeouts
- Hypothesis 5: SQL problems

4.2.1 Lock waits due to the default LOCKTIMEOUT parameter value

Lock waits are a common and natural occurrence in any multi-user application environment where the same data is shared. It is only when such lock waits occur far too frequently and involve extended delays that it becomes a performance problem. In this scenario, the lock wait problems are caused by the default LOCKTIMEOUT parameter value of -1, which causes an application to wait forever for a currently unavailable lock. System defaults are often accepted in the real world and can lead to the kinds of problems described here.

Description of the application

To simulate the lock wait situation, we used the Trade2 application in conjunction with the two following command line applications, neither of which issued commits:

- Application 1 updates the rows of one of the TRADEQUOTEBEAN table without issuing commits (**db2 +c**), as shown in Figure 4-3.

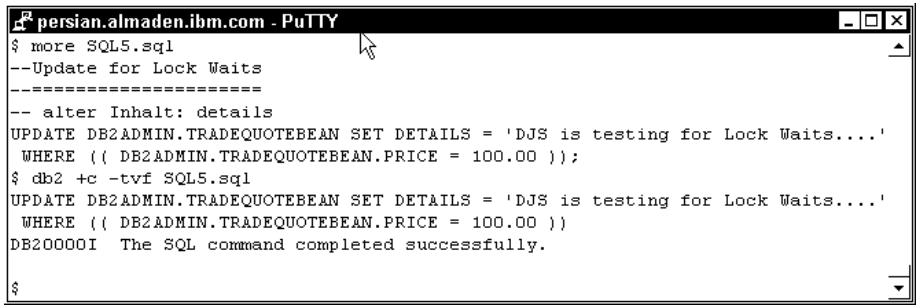


Figure 4-3 Application 1 with update statement for the LOCKTIMEOUT scenario

- Application 2 selects the rows being updated by Application 1, as shown in Figure 4-4 on page 68.

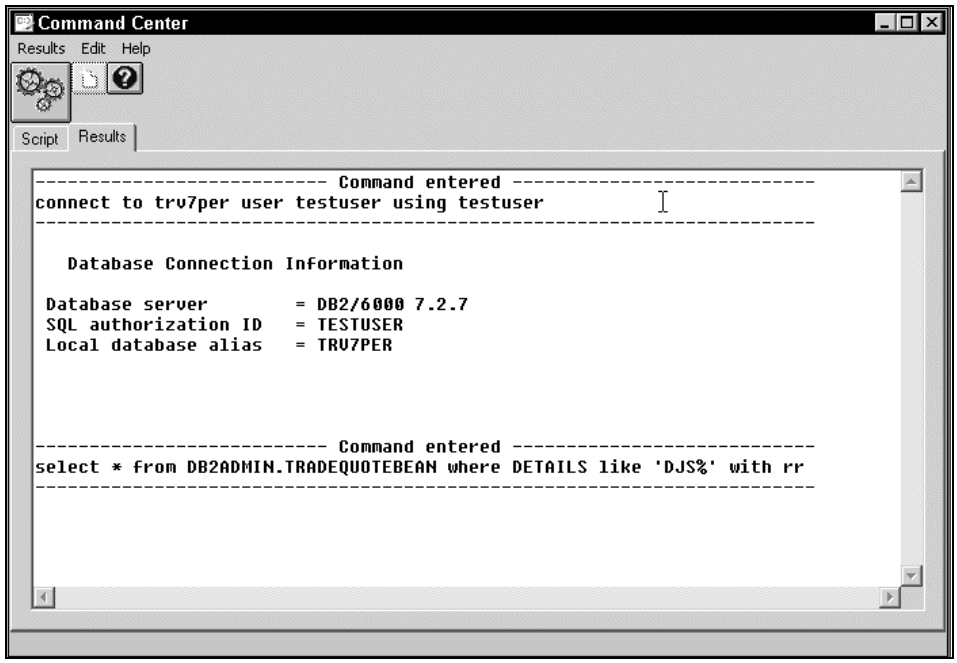


Figure 4-4 Application 2 with select statement for the LOCKTIMEOUT scenario

Environment configuration

Figure 4-5 on page 69 shows the environment used for the lock waits scenario. TRADEDB and the DB2 PE Server are installed on PERSIAN, which has DB2 UDB V7.2 installed. DB2 PE Clients are NALUR6 and NALUR8.

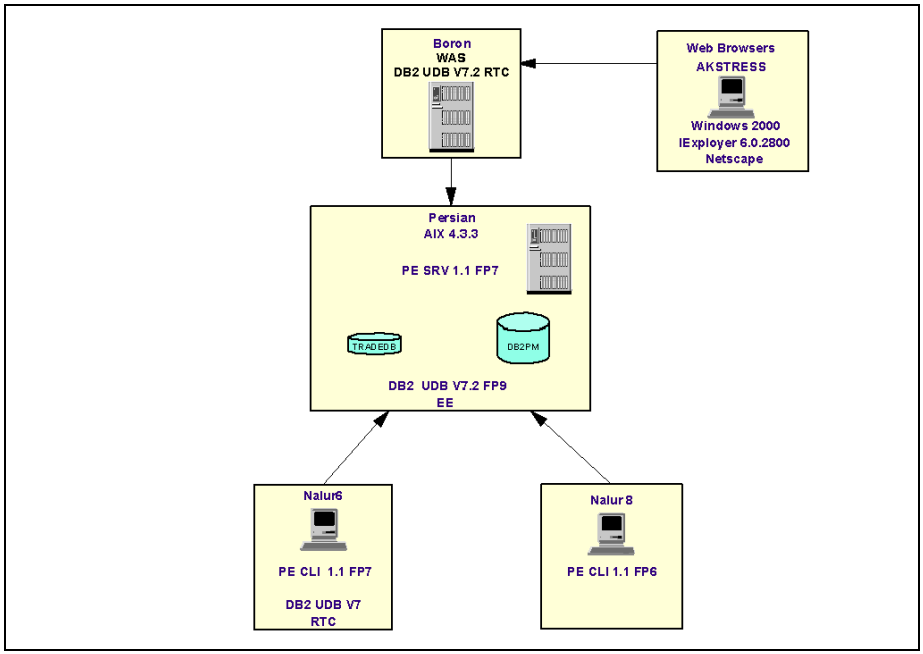


Figure 4-5 Lock waits scenario environment

Monitor level settings

The only changes we made for this specific scenario were to the auto refresh interval for the Application Summary, which we set to 15 seconds.

Workload used

The AKSTRESS tool was used to simulate 200 concurrent user on AIX and 20 concurrent user on Windows performing login, stock trading and logout tasks.

We wrote batch files with custom SQL statements shown in Figure 4-3 on page 67 and Figure 4-4 on page 68, and executed them using four concurrent command windows (two for each application).

Triggering event

Several users complained about not getting any response from the system, while others complained about receiving application error messages asking them to try accessing the application again later. Complaints received at the help desk are typically channeled to application support, systems operations and the DBA as appropriate to resolve the problem. The problem determination process is then initiated as described in Figure 4-1 on page 62. We concluded that the problems

reported were having a significant negative impact on the business, and needed to be resolved promptly. We therefore had to diagnose the problem with minimal exception monitoring, and resolve it quickly.

Hypotheses and their validation

We postulated the hypotheses described in “Hypotheses and their validation” on page 66 as the potential cause of the problem. Each of these hypotheses was validated in turn using information available from the WebSphere Application Server, DB2 logs and DB2 Performance Expert.

Hypothesis 1: Resource constraints

Given our controlled environment, we ignored real world root cause possibilities such as network bandwidth concerns, Web server problems, system utilization, and process priorities.

Hypothesis 2: WebSphere Application Server problems

Since the user’s trading application was accessed using the Web and the WebSphere Application Server, we checked to see whether there were any bottlenecks in the WebSphere Application Server environment. A view of the WebSphere Advanced Administrative Console showed a number of `Timed out waiting for a connection from data source jdbc/TRADEDB...` messages, as shown in Figure 4-6.

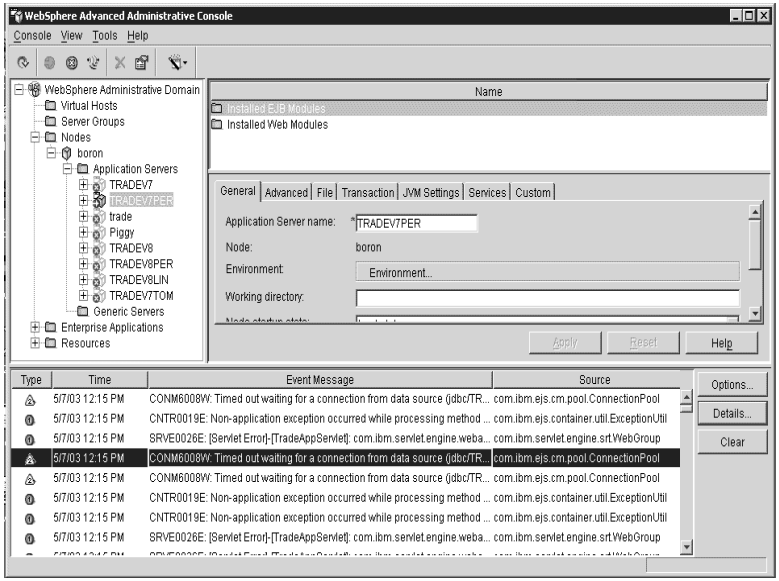


Figure 4-6 WebSphere Application Server - Connection Timed out warnings

When the application is unable to get a connection to the TRADEDB, it displays a window asking the user to try the application again later. We have therefore identified at least one of the causes of the problems reported by the user.

We then checked the Resource Analyzer as shown in Figure 4-7 and noticed that the Average Pool Size was 10, which is the default maximum value of the connection pool size. Given that our workload involved 20 concurrent users, the inadequate size of the connection pool was certainly the cause of the application errors.

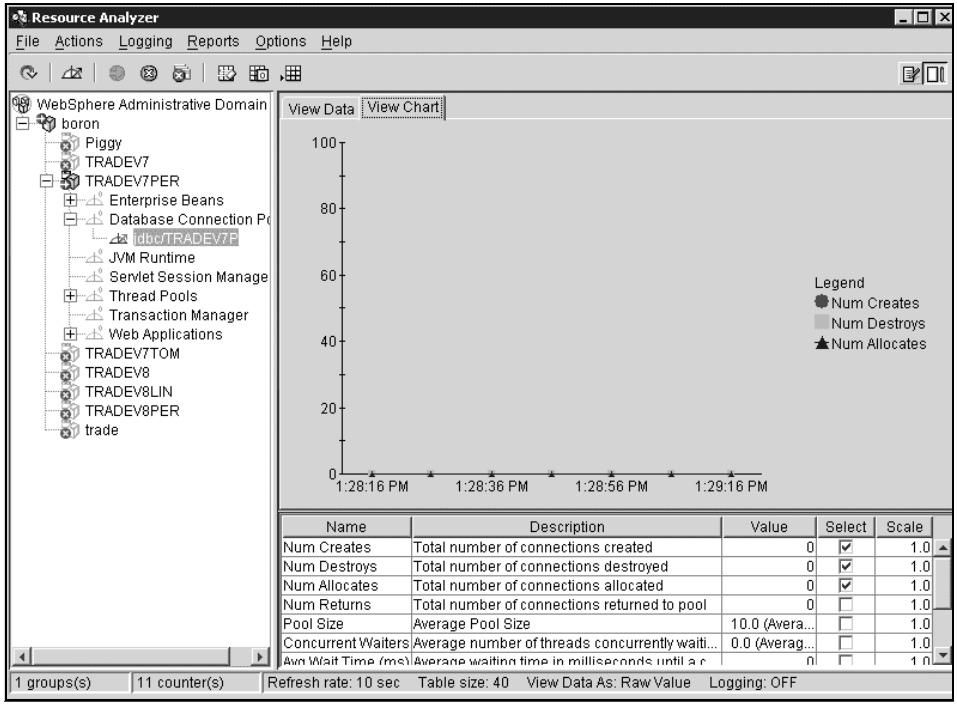


Figure 4-7 Resource Analyzer - Database Connection Pool

Note: While the inadequate connection pool size was a cause of one of the problems encountered by the users, it did not explain the lack of system response encountered by some of the users. We therefore decided to probe further into DB2 to look for the causes of the extended waits for a response.

Hypothesis 3: Maximum users/connections thresholds exceeded

We checked to see whether the maximum allowable number of active applications (MAXAPPLS)[1] had been reached for this database.

The Statistics Details window of DB2 PE shown in Figure 4-8 indicated the maximum concurrent connections to be 14, which is below the maximum active applications (MAXAPPLS) value of 20 shown in Figure 4-9 on page 73.

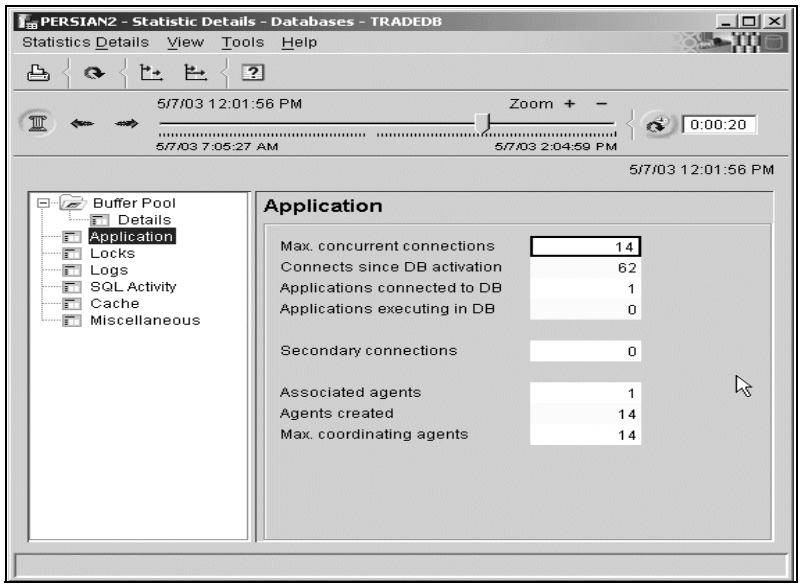


Figure 4-8 Statistics Details: maximum concurrent connections

[1] When this threshold is reached, an SQL error code -1040 is returned to the application, which should then take appropriate action, for example, by sending a message to the user requesting that (s)he retry the application at a later time. The error message for SQL1040N in the *IBM DB2 UDB Message Reference Volume 2* (GC09-4841-00) states that *"the maximum number of applications is already connected to the database"*, and the user response should be "wait for other applications to disconnect from the database. If more applications are required to run concurrently, increase the value for maxappls. After all applications disconnect from the database and the database is restarted, the new value takes effect.".

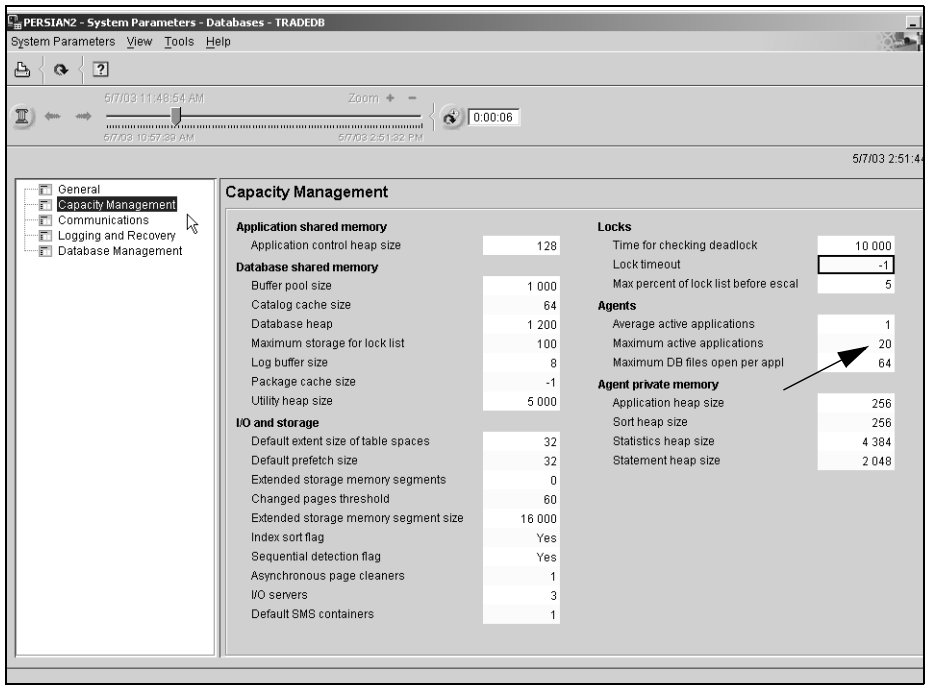


Figure 4-9 System Parameters - Database TRADEDB - Capacity Management

Note: This clearly indicated that the application errors were not caused by an undersized MAXAPPLS. We therefore decided to probe further.

One may also check the application log of WebSphere Application Server to determine whether there are any problems connecting to the database. We have not shown the contents of this file here.

Important: We will return to the highlighted *Lock timeout* value of -1 shortly.

Hypothesis 4: Locking problems-- waits, deadlocks, timeouts

Since lock waits could be the source of the extended waits, we decided to check them using DB2 PE.

We selected the **TRADEDB** database in the Statistics Details window of DB2 PE, and enabled history so that we could move back in time to display the window shown in Figure 4-10 on page 74. The reason for moving into history mode was to be able to view the state of the system when the problem was said to be occurring, since the current state may not help in problem diagnosis.

.

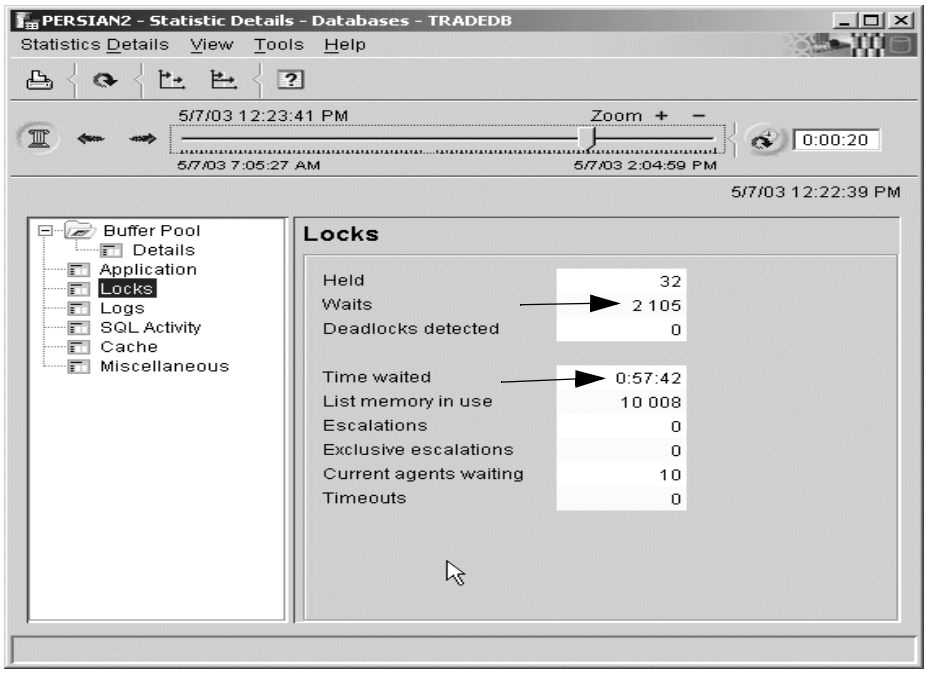


Figure 4-10 Statistics Details of TRADEDB in history mode

We observed that there were many lock waits on this database and the time waited for the locks was also very high. We also noticed that there were no escalations or time-outs.

The lack of time-outs led us to look up the LOCKTIMEOUT parameter value in the Systems Parameter window by clicking **System Overview -> System Parameters -> Databases -> Capacity Management.** Figure 4-9 on page 73 shows that the Lock timeout parameter had the default value of -1, which specifies that lock time-out detection should be disabled, that is, a lock requester should wait indefinitely for the lock request to be satisfied.

Root cause of the problem

The cause of the “try again later” application error messages was the inadequate default value of 10 for the maximum size of the connection pool in WebSphere Application Server.

The cause of the extended waits was the default Lock timeout value of -1.

Application of best practices

The defaults for both the WebSphere Application Server maximum connection pool size and the LOCKTIMEOUT have to be changed to address the specific needs of the application.

Appropriate monitoring of the WebSphere Application Server using Resource Analyzer will provide guidelines on the Average Pool Size (see Figure 4-7 on page 71), which can then be suitably adjusted to arrive at a maximum connection pool size. Ongoing monitoring and adjustments will be required to ensure that this parameter will not cause performance bottlenecks.

The desired value of LOCKTIMEOUT would depend upon the nature of the workload. Setting too low a value for this parameter can result in spurious time-outs, thereby frustrating users, while a very large value can cause extended waits that would also be unacceptable to users. Typical values for an OLTP workload are between 10 and 15 seconds, while 60 seconds may be an appropriate value for BI/DW environments. Here again, ongoing monitoring will help you to determine the appropriate value in your environment.

Important: A LOCKTIMEOUT setting of 0 is *not* recommended, because this disables lock waiting; if a lock is not available when requested, DB2 immediately returns a -911 code to the application.

If the average lock wait times are consistently high, consider tuning the applications that hold a large number of locks, to reduce the size and duration of the locks held.

DB2 PE provides a number of windows to determine current locking conflicts in applications, and the participants (lock waiter and lock holder) in the locking conflict. The main windows are shown in Figure 4-11 on page 76, Figure 4-12 on page 77, Figure 4-13 on page 77 and Figure 4-14 on page 78.

Figure 4-11 on page 76 shows portions of the Application Summary, Application in Lock Conflicts (Lock Type table lock type), and Locking Conflicts.

- The Application Summary window shows a lock wait state on one of the applications.
- The Application in Locking Conflicts window lists the applications involved in the lock conflict. The window shows two applications involved in the conflict: the lock holder and the lock waiter. The lock mode for both applications is Share.
- The Locking Conflicts window shows the table that is locked, and the lock type used, which is a Share lock.

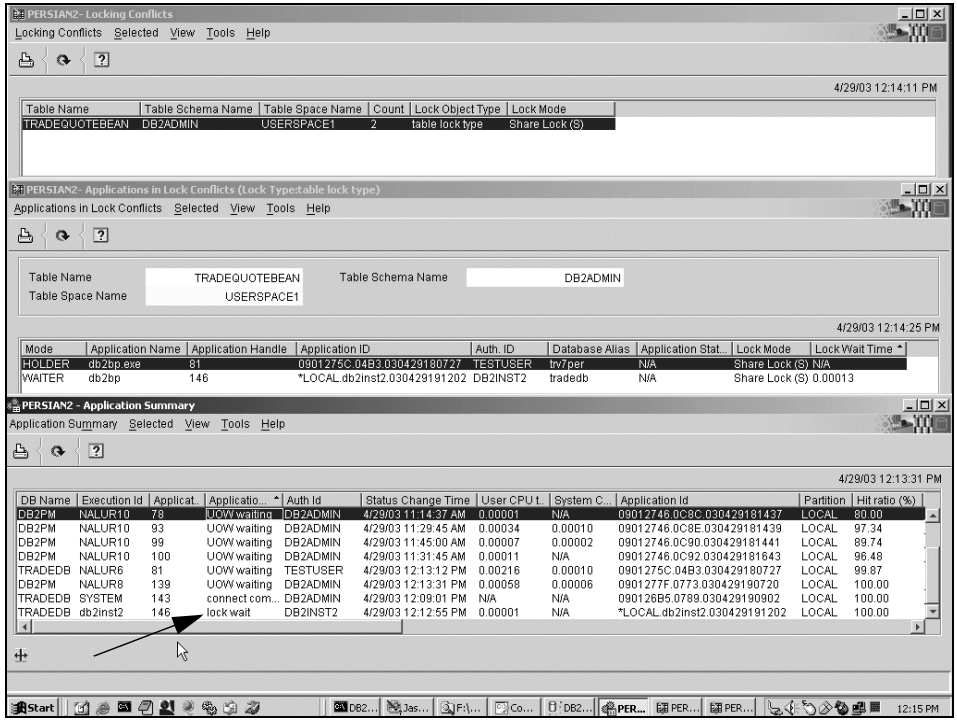


Figure 4-11 Application summary, Applications in Locking Conflicts, etc.

All the windows shown in Figure 4-11 can be used to identify a current lock wait situation. To obtain additional information about the lock waiter or the lock holder, you need to double-click the application in the Application Summary window to see the Application Detail window as shown in Figure 4-12 on page 77.

The SQL Statement and Package selection in the Application Detail window shows the SQL statement involved in the conflict, which happens to be the lock holder with a SELECT SQL statement. The waiter (window not shown here) is an SQL UPDATE statement.

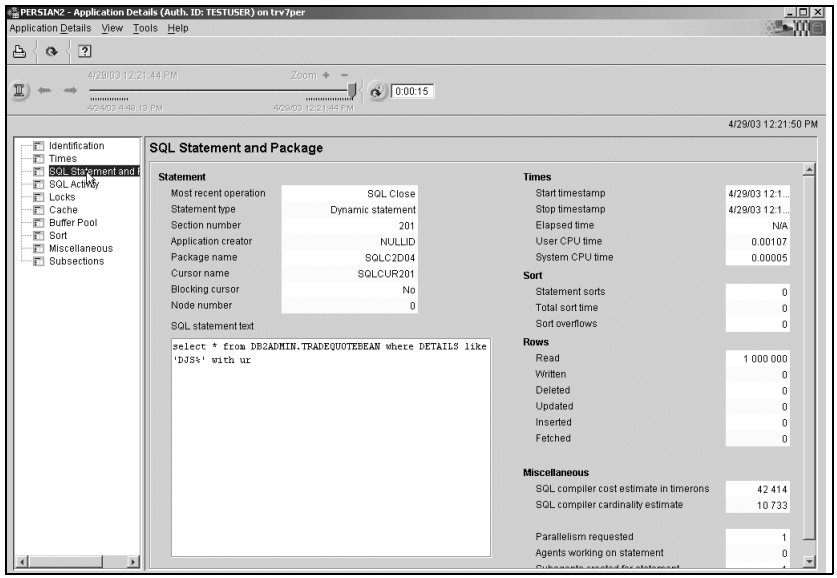


Figure 4-12 Application Details - SQL of lock holder

One may obtain more details about the locks held by an application by selecting the **Locks** option in the Application Details window, as shown in Figure 4-13.

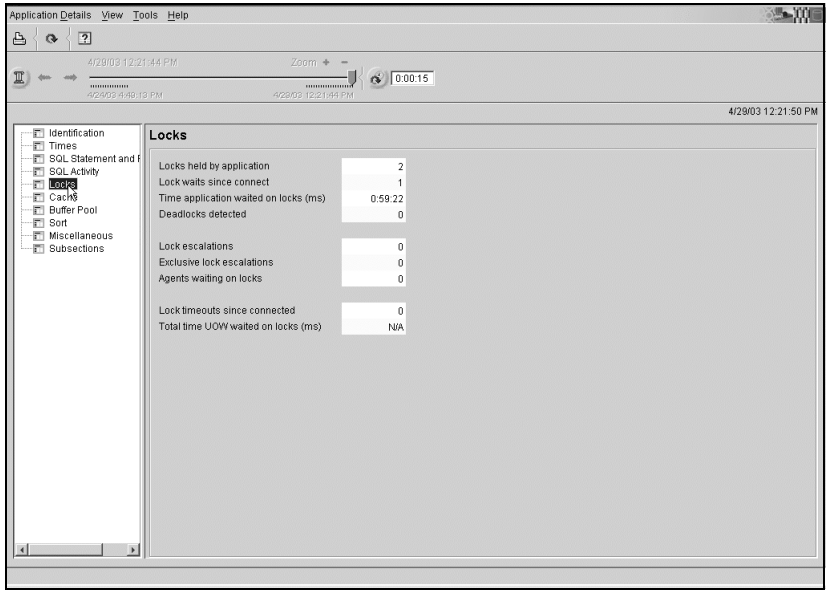


Figure 4-13 Application Details - Locks of lock holder

This window shows the number of escalations, time-outs and locks held by the application. Figure 4-13 on page 77 shows the application holding two locks, with no time-outs or escalations. Figure 4-11 on page 76, Figure 4-12 on page 77 and Figure 4-13 on page 77 display the current state of the system.

As described earlier, the DBA sometimes needs to look at the state of the system at a prior point in time when a problem is known to have occurred. DB2 PE's history capability allows the DBA to scroll back in time as shown in Figure 4-14.

Since the application was known to have accessed the TRADEDB database, we enabled history in the Statistics Details window and selected **TRADEDB** after double-clicking **Databases**.

.

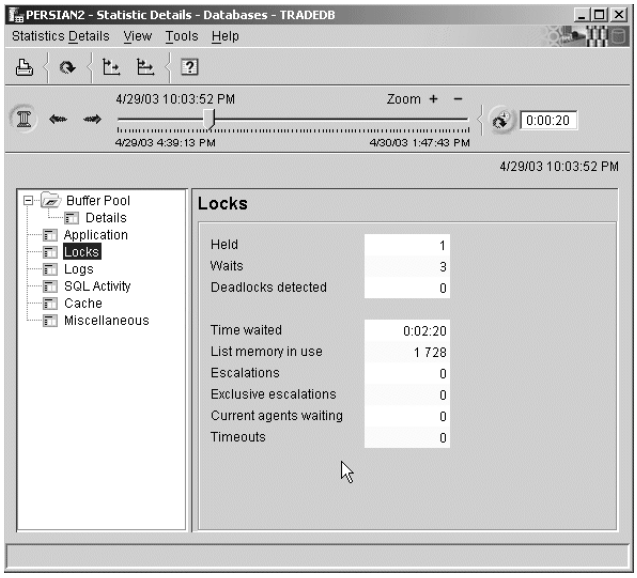


Figure 4-14 Statistics Details of TRADEDB in history mode

Figure 4-14 shows that there were some lock waits since resetting the monitor switches and starting the PE server. Here again, we observed the lack of time-outs.

Note: Certain DB2 configuration parameters do not take effect until the DB2 instance has been restarted, or all connections have ceased against the database, or the dbm has been restarted. The System Parameters windows in DB2 Performance Expert may show you the current configuration value setting, but this may not be the one currently being used by the database.

4.2.2 Lock waits, deadlocks and time-outs due to lock escalations

The scenario shown here is one where the database configuration parameter LOCKLIST value is undersized for the given workload, thereby resulting in lock escalation, which in turn results in poor user response times deadlocks and frequent time-outs.

The database configuration parameter LOCKLIST specifies the maximum amount of memory that may be used for locks by all the current applications, while the database configuration parameter MAXLOCKS specifies the maximum percentage of LOCKLIST that may be consumed by a single application. When either of these thresholds are exceeded, DB2 attempts to alleviate the problem by escalating row level locks[2] on tables to a table level lock, a process known as *lock escalation*. While constrained memory is relieved, the side effect of this action is potentially reduced concurrency, which manifests itself as time-outs, deadlocks and extended response times. Lock escalations may be exclusive or shared, with the exclusive variety being the more restrictive in terms of concurrency. Therefore, lock escalations are undesirable events.

Important: In order to force lock escalations to occur in our controlled environment, we deliberately set the LOCKLIST to a very small value.

This scenario is a follow-on of the previous scenario described in 4.2.1, “Lock waits due to the default LOCKTIMEOUT parameter value” on page 66.

Note: The LOCKTIMEOUT parameter value was set to 10 seconds for this scenario, and we increased the WebSphere Application Server maximum connection pool size to 40.

Description of the application

This application is identical to the one described in “Description of the application” on page 67.

Environment configuration

This is identical to the one described in “Environment configuration” on page 68.

Monitor level settings

These are identical to the ones described in “Monitor level settings” on page 69.

[2] Each lock requires 72 bytes of memory for an object that has no other locks held on it, or 36 bytes of memory for an object that has existing locks held on it. If a number of row locks can be replaced with a single table lock, the locking storage area can be used by other applications.

Workload used

The workload used was identical to the one described in “Workload used” on page 69.

Triggering event

Our triggering event in this case was user complaints about extended response times as well as frequent time-outs.

Hypotheses and their validation

We postulated the hypotheses described in “Hypotheses and their validation” on page 66 as the potential cause of the problem. Each of these hypotheses was validated in turn using information available from the WebSphere Application Server, DB2 logs and DB2 Performance Expert.

Hypothesis 1: Resource constraints

Given our controlled environment, we ignored real world root cause possibilities such as network bandwidth concerns, Web server problems, system utilization, and process priorities.

Hypothesis 2: WebSphere Application Server problems

Using the approach described in “Hypothesis 2: WebSphere Application Server problems” on page 70, we concluded that the WebSphere Application Server was not the cause of our problems.

Hypothesis 3: Maximum users/connections thresholds exceeded

Using the approach described in “Hypothesis 3: Maximum users/connections thresholds exceeded” on page 72, we concluded that MAXAPPLS was not the cause of our problems.

Hypothesis 4: Locking problems - waits, deadlocks, timeouts

Using the approach described in “Hypothesis 4: Locking problems-- waits, deadlocks, timeouts” on page 73, we discovered that exclusive lock escalations had occurred along with a large number of time-outs as shown in the Statistics Details window for the TRADEDB Locks option in Figure 4-15 on page 81.

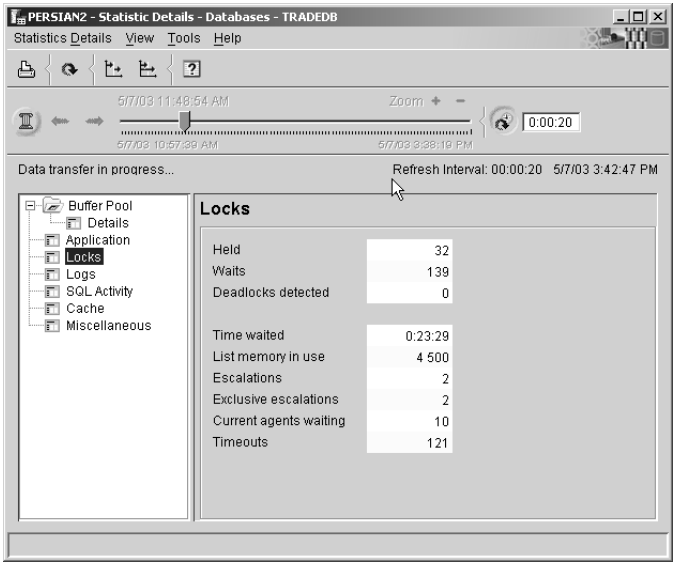


Figure 4-15 Statistics Details of TRADEDB shows lock escalations

Note: One can determine the applications affected by these lock escalations by viewing the Applications in the Lock Conflicts window as discussed when describing the windows in Figure 4-11 on page 76. With lock escalation, you will discover an exclusive lock based on table mode.

To determine the cause of the lock escalation, you need to access the contents of the *db2diag.log* which will provide details of the statement, provided the DIAGLEVEL is set to 4. Remember that we set the DIAGLEVEL to 4 in our scenarios for the purpose of problem diagnosis. In the real world, you will need to set this level as an exception monitoring exercise for an appropriate duration while the problems are being experienced.

Figure 4-16 shows the content of the *db2diag.log*. It highlights the SQL statement that triggered the escalation, but does not identify whether it occurred due to the LOCKLIST or MAXLOCKS threshold being exceeded.

Note: In DB2 UDB V8, lock escalation information can also be found in the DB2 Administration notification file (*db2inst2.nfy* in our environment) located in the same directory as the *db2diag.log* file.

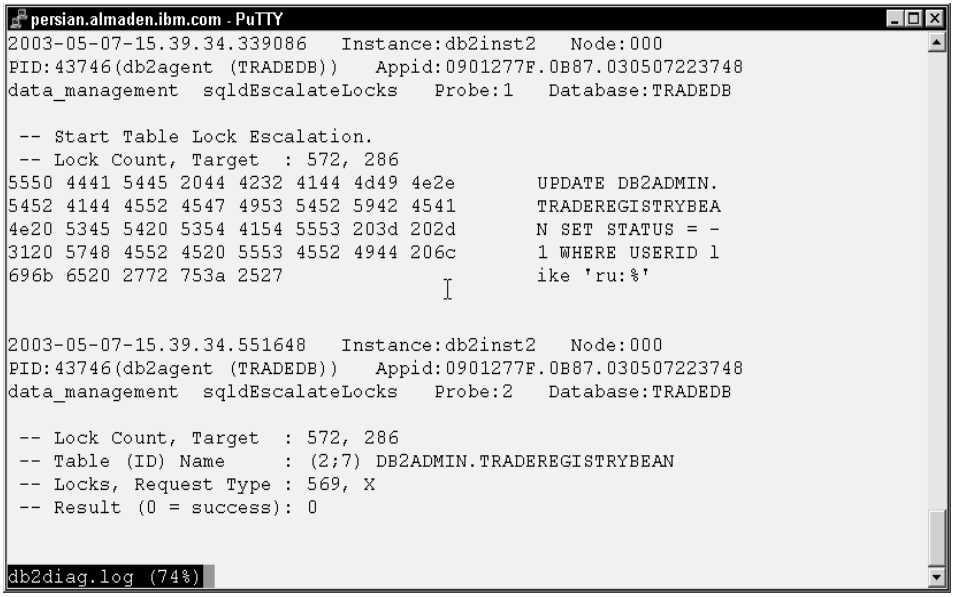


Figure 4-16 db2diag.log file

Root cause of the problem

While it is clear that the SQL UPDATE statement triggered the escalation, it is not obvious whether LOCKLIST or MAXLOCKS is the culprit.

The values for these parameters can be determined by looking up the System Parameters window for the TRADEDB as shown in Figure 4-17 on page 83; LOCKLIST is 120 4K pages, while MAXLOCKS is 50%.

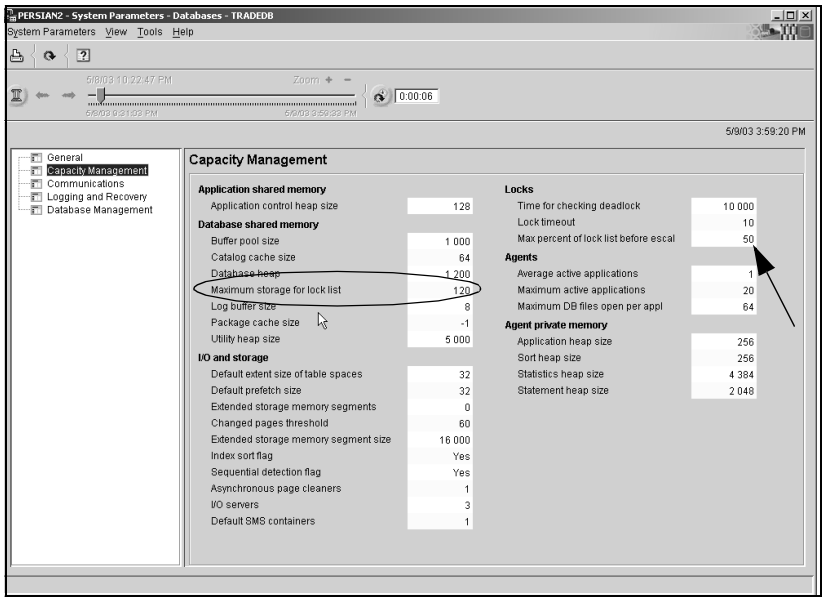


Figure 4-17 LOCKLIST and MAXLOCKS values

Application of best practices

The default values for most configuration settings are probably not appropriate for every implementation of DB2.

One needs to determine the maximum number of locks held by any single application, the maximum concurrent applications, and the memory consumed per lock to determine the LOCKLIST size and the value for MAXLOCKS.

The value for maximum concurrent applications can be the MAXAPPLS value. However, when this value is specified as automatic, one needs to determine the maximum number of concurrent applications at peak activity using the Statistics Details window for this database, as shown in Figure 4-8 on page 72.

The maximum concurrent locks held by any single application can be determined by creating an Event Monitor that collects transaction events. However, we do not provide an example of how to collect this information.

Refer to the *IBM DB2 Administration Guide: Performance*, SC09-4821 for details on computing an appropriate value for LOCKLIST and MAXLOCKS.

4.2.3 Long running SQL

The scenario shown here is one of diagnosing the performance problem of a specific long running SQL query executed for the very first time. The SQL query in question was still executing when we performed our problem diagnosis.

The reasons for poor SQL performance can range from resource constraints (CPU, memory, I/O), sub-optimal access paths, inadequate database housekeeping (reorgs and runstats), to inefficient coding. In typical performance tuning, modifying the SQL is the least desirable activity even though, in practice, inefficient coding may account for the majority of the performance problems encountered.

Description of the application

We ran the Trade2 application without the batch jobs and executed a couple of queries in parallel. One of these queries was the problematic long running SQL query.

Environment configuration

This is identical to the one described in “Environment configuration” on page 68.

Monitor level settings

This is identical to the one described in “Monitor level settings” on page 65.

Workload used

The IBM AKSTRESS tool was used to simulate 200 concurrent users, with two other queries, one of which was the problematic long running SQL query.

Triggering event

Our triggering event in this case was a customer call requesting assistance with diagnosing the poor performance of a currently executing query that the user had initiated from NALUR6. In the course of the discussion with the user, it was revealed that the query was being executed for the first time ever.

Hypotheses and their validation

We postulated the hypotheses described in “Hypotheses and their validation” on page 66 as the potential cause of the problem, with the exception of WebSphere Application Server problems, since the problem query was not being executed through it. Each of these hypotheses was validated in turn using information available from the DB2 logs, Control Center, and DB2 Performance Expert.

Hypothesis 1: Resource constraints

Given our controlled environment, we ignored real world root cause possibilities such as network bandwidth concerns, Web server problems, system utilization, and process priorities.

Hypothesis 2: WebSphere Application Server problems

This hypothesis is not applicable in this scenario.

Hypothesis 3: Maximum users/connections thresholds exceeded

Since the user had managed to connect to the database in order to issue the query, this hypothesis is not applicable either.

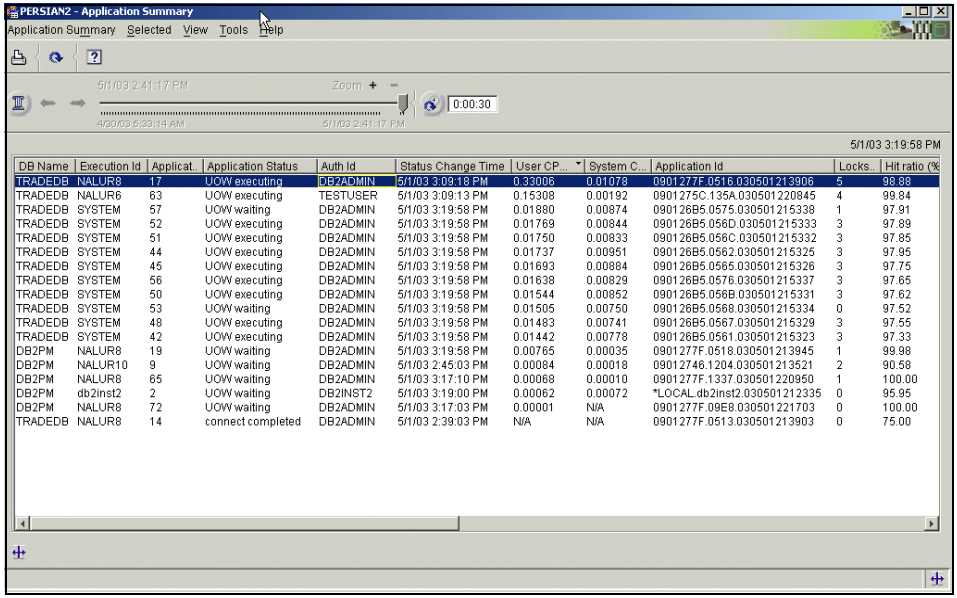
Hypothesis 4: Locking problems - waits, deadlocks, timeouts

Using the same approach described in “Hypothesis 4: Locking problems-- waits, deadlocks, timeouts” on page 73, we discovered that there were no extended waits occurring and therefore ruled out locking problems as the cause of the performance problem.

Hypothesis 5: SQL problems

We focused on probing information in the Application Summary window shown in Figure 4-18. Not too many concurrent applications are executing, and the specific application relating to NALUR6 indicates high CPU consumption.

.



DB Name	Execution Id	Applicat..	Application Status	Auth Id	Status Change Time	User CP...	System C...	Application Id	Locks..	Hit ratio (%
TRADEDB	NALUR8	17	UOW executing	DB2ADMIN	5/1/03 3:09:18 PM	0.33006	0.01078	0901277F.0516.030501213906	5	98.88
TRADEDB	NALUR6	63	UOW executing	TESTUSER	5/1/03 3:09:13 PM	0.15308	0.00192	0901275C.135A.030501220845	4	99.84
TRADEDB	SYSTEM	57	UOW waiting	DB2ADMIN	5/1/03 3:19:58 PM	0.01880	0.00874	090126B5.0575.030501215338	1	97.91
TRADEDB	SYSTEM	52	UOW executing	DB2ADMIN	5/1/03 3:19:58 PM	0.01769	0.00844	090126B5.056D.030501215333	3	97.89
TRADEDB	SYSTEM	51	UOW executing	DB2ADMIN	5/1/03 3:19:58 PM	0.01750	0.00833	090126B5.056C.030501215332	3	97.85
TRADEDB	SYSTEM	44	UOW executing	DB2ADMIN	5/1/03 3:19:58 PM	0.01737	0.00951	090126B5.0562.030501215325	3	97.95
TRADEDB	SYSTEM	45	UOW executing	DB2ADMIN	5/1/03 3:19:58 PM	0.01693	0.00884	090126B5.0565.030501215326	3	97.75
TRADEDB	SYSTEM	56	UOW executing	DB2ADMIN	5/1/03 3:19:58 PM	0.01638	0.00829	090126B5.0576.030501215337	3	97.65
TRADEDB	SYSTEM	50	UOW executing	DB2ADMIN	5/1/03 3:19:58 PM	0.01544	0.00852	090126B5.056B.030501215331	3	97.62
TRADEDB	SYSTEM	53	UOW waiting	DB2ADMIN	5/1/03 3:19:58 PM	0.01505	0.00750	090126B5.0568.030501215334	0	97.52
TRADEDB	SYSTEM	48	UOW executing	DB2ADMIN	5/1/03 3:19:58 PM	0.01483	0.00741	090126B5.0567.030501215329	3	97.55
TRADEDB	SYSTEM	42	UOW executing	DB2ADMIN	5/1/03 3:19:58 PM	0.01442	0.00778	090126B5.0561.030501215323	3	97.33
DB2PM	NALUR8	19	UOW waiting	DB2ADMIN	5/1/03 3:19:58 PM	0.00765	0.00035	0901277F.0518.030501213945	1	99.98
DB2PM	NALUR10	9	UOW waiting	DB2ADMIN	5/1/03 2:45:03 PM	0.00084	0.00018	09012746.1204.030501213521	2	90.58
DB2PM	NALUR8	65	UOW waiting	DB2ADMIN	5/1/03 3:17:10 PM	0.00068	0.00010	0901277F.1337.030501220950	1	100.00
DB2PM	db2inst2	2	UOW waiting	DB2INST2	5/1/03 3:19:00 PM	0.00062	0.00072	*LOCAL.db2inst2.030501213335	0	95.95
DB2PM	NALUR8	72	UOW waiting	DB2ADMIN	5/1/03 3:17:03 PM	0.00001	N/A	0901277F.09E8.030501221703	0	100.00
TRADEDB	NALUR8	14	connect completed	DB2ADMIN	5/1/03 2:39:03 PM	N/A	N/A	0901277F.0513.030501213903	0	75.00

Figure 4-18 Application Summary: long running SQL

Since the causes of high CPU usage can depend on various factors, we needed to dig further into the details of the application by double-clicking the **NALUR6** application row. Selecting the **SQL Statement and Package** option in the Application Details window brings up the windows shown in Figure 4-19.

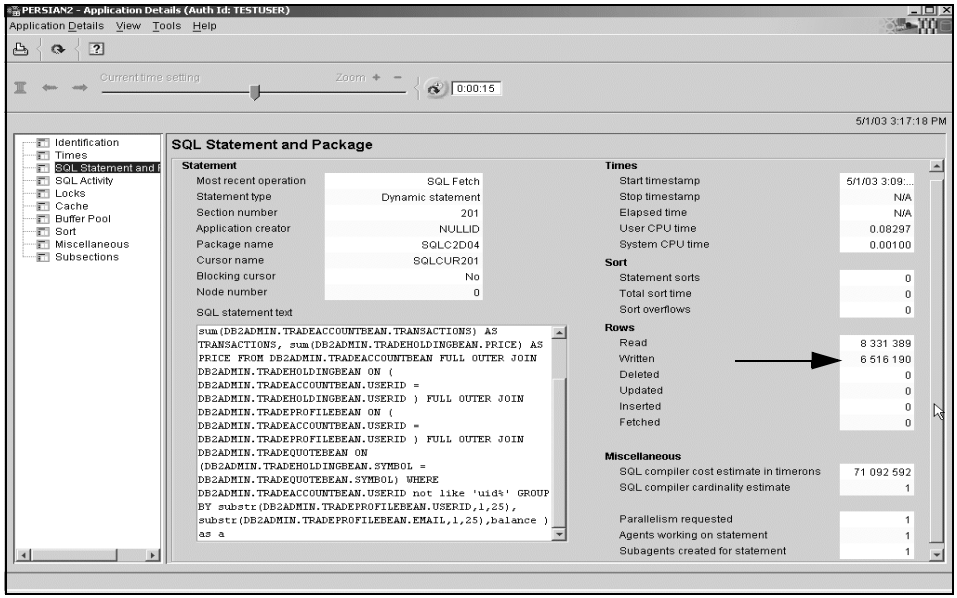


Figure 4-19 Application Details - long running SQL

The Rows section indicates more than 6 million rows being written, even though the SQL statement text shows a SELECT statement which is a read-only operation. These writes correspond to rows being written into the tempspace, which is used for sort operations in DB2.

A review of the SQL statement text also showed full outer joins which can be very expensive in terms of performance. We needed to investigate the SQL statement further.

We therefore chose to use DB2's EXPLAIN facility through the Command Center to obtain details of the access path chosen by DB2 for executing this query.

Important: While DB2 Performance Expert for z/OS supports direct invocation of the EXPLAIN feature, this functionality is currently not available in the Multiplatform product.

The results of the EXPLAIN of this query, shown in Figure 4-20, clearly indicate table scans rather than index access.

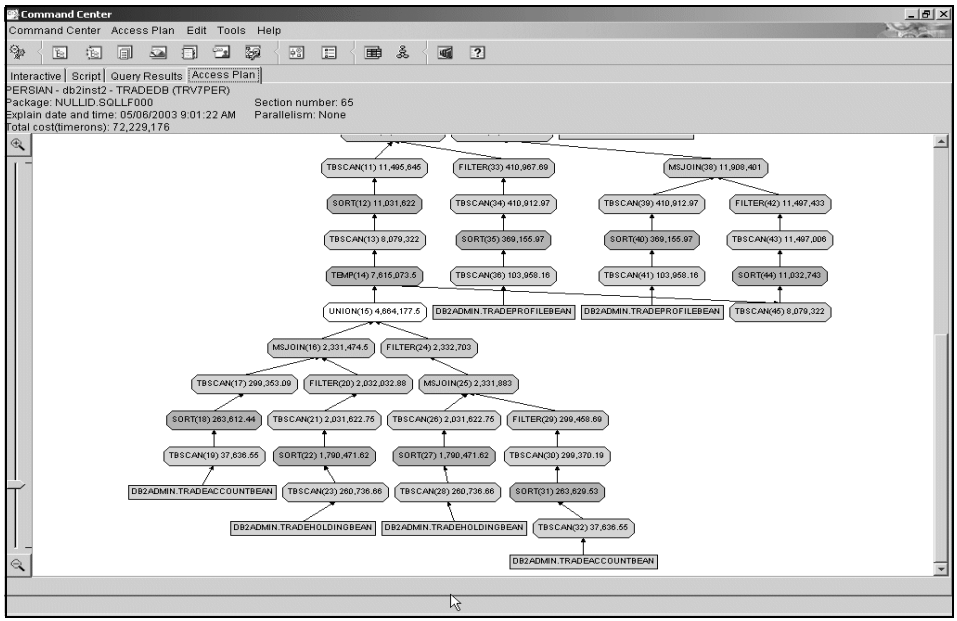


Figure 4-20 Access plan of the long running SQL

While table scans by themselves do not necessarily indicate a performance problem, their presence should set off alarm bells, especially when joins are involved.

We therefore decided to check whether indexes had been defined at all, and if so, why the indexes had not been chosen as access paths.

We used the Control Center to ascertain that a primary index had been defined for each of the tables in the trade application, as shown in Figure 4-21.

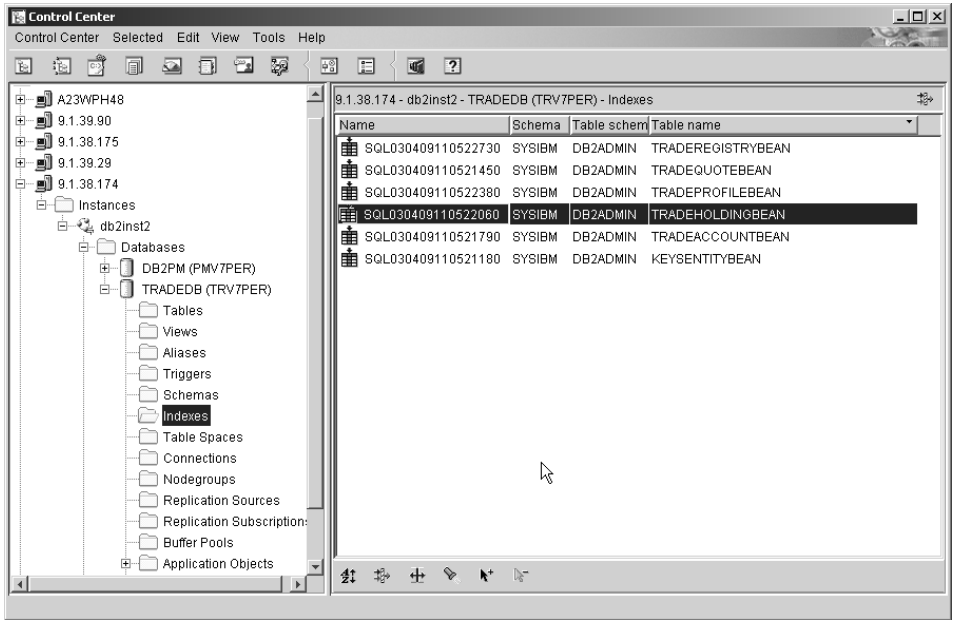


Figure 4-21 Control Center - Indexes of TRADEDB

Further details about the index can be obtained by double-clicking the appropriate index name, as shown in Figure 4-22.

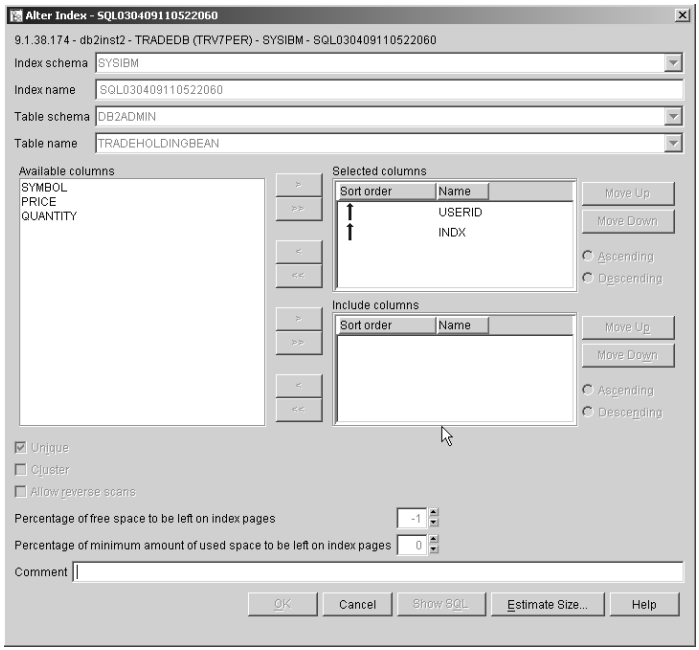


Figure 4-22 Control Center - Index definition

We saved the query in a file name *sq5.sql* and called the **db2advis** tool on AIX to get its recommendation for indexes on the tables involved in the query, as shown in Figure 4-23. The tool came up with no recommendations for indexes for this query.

```
persian.almaden.ibm.com - PuTTY
$ db2advis -d tradedb -p -i sq5.sql
execution started at timestamp 2003-05-06-10.22.05.653783
  found [1] SQL statements from the input file

Calculating initial cost (without recommmended indexes) [72229176.000000] timer
ns
Initial set of proposed indexes is ready.
Found maximum set of [0] recommended indexes
Cost of workload with all indexes included [72229176.000000] timerons
total disk space needed for initial set [   0.000] MB
total disk space constrained to         [  -1.000] MB
  0  indexes in current solution
 [72229176.0000] timerons  (without indexes)
 [72229176.0000] timerons  (with current solution)
 [%0.00] improvement

 Trying variations of the solution set.
--
-- execution finished at timestamp 2003-05-06-10.22.26.230341
--
--
-- LIST OF RECOMMENDED INDEXES
-- ===========================
--  no indexes are recommended for this workload.
-- ===========================
--
Index Advisor tool is finished.
```

Figure 4-23 Index advisory tool on AIX

We then decided to look at the statistics on the tables used in the SQL query using the command window, as shown in Figure 4-24. The results showed that **runstats** had not been collected for the tables used in the query, which tends to result in suboptimal access path selection.

```
9.1.38.174 - PuTTY

db2inst1@malmo [/usr/doreen] # db2 -v "select substr(tabschema,1,20), substr(>
select substr(tabschema,1,20), substr(tabname,1,20), card, npages, fpages from syscat.tables
re tabschema = 'DB2ADMIN'

1                    2                    CARD                 NPAGES      FPAGES
-------------------- -------------------- -------------------- ----------- -----------
DB2ADMIN             KEYSENTITYBEAN                         -1          -1          -1
DB2ADMIN             TRADEACCOUNTBEAN                       -1          -1          -1
DB2ADMIN             TRADEHOLDINGBEAN                       -1          -1          -1
DB2ADMIN             TRADEPROFILEBEAN                       -1          -1          -1
DB2ADMIN             TRADEQUOTEBEAN                         -1          -1          -1
DB2ADMIN             TRADEREGISTRYBEAN                      -1          -1          -1

  6 record(s) selected.

$
```

Figure 4-24 Statistic information of TRADEDB tables

Root cause of the problem

It appears that the problem is the absence of actual statistics on the tables used in the query, resulting in DB2 generating a suboptimal access plan for the query.

Application of best practices

We ran **runstats** on the tables involved in the query (not shown here), and displayed the updated statistics as shown in Figure 4-25.

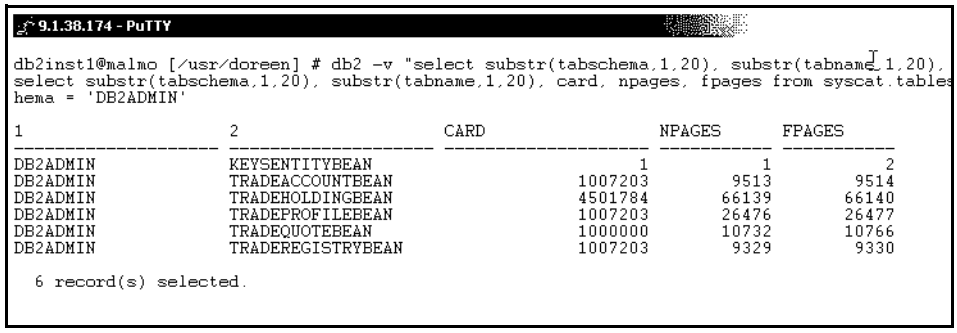


Figure 4-25 Statistic information of TRADEDB tables, after runstats

We then reran EXPLAIN to display the selection of the primary indexes in the generated access plan, as shown in Figure 4-26 on page 92.

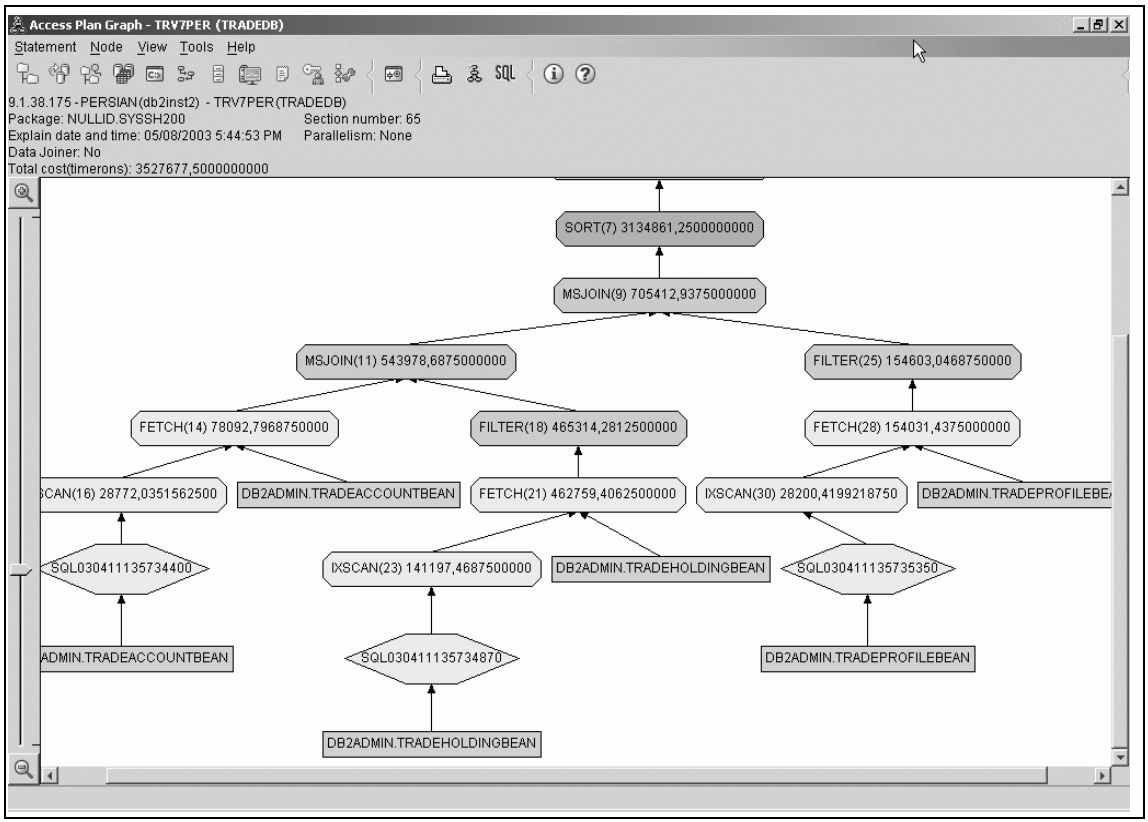


Figure 4-26 Access plan after running statistics

A rerun of the query with the revised access plan resolved the performance problems (this is not shown here).

4.3 Online/realtime event monitoring scenarios

As discussed in 1.2.2, “Online/realtime event monitoring” on page 5, online/realtime event monitoring involves looking out for specific events that may either identify a specific problem or portend problems in the near to immediate future, in order to take prompt corrective action.

The events may be “one of” such as a database becoming unavailable or a file system becoming full, or “short term trends” such as a burst of deadlocks, time-outs and lock escalations, or a sharp increase in the number of units of work.

Such monitoring is essentially preemptive in nature, where the DBA attempts to forestall and react to potential availability and performance problems before user complaints begin pouring in. This requires the DBA to leverage his knowledge about the application workload in the target environment to set up custom monitoring events and thresholds.

DB2 PE's System Health functionality described in "System Health" on page 30 enables a DBA to define custom data views for a wide range of performance-related counters such as units of work, hit ratios, lock escalations, and sort overflow percentages. Data views include graphs as well as raw data, and filters can be specified to limit displayed data to counter values that only exceed user-defined thresholds.

Note: Data views do not support triggering automatic corrective action or alerts to DBAs when certain events occur, or when specific counters reach user-defined thresholds. Data views' thresholds are meant to filter what is displayed.

Important: DB2 PE for z/OS provides an exception processing mechanism that can alert the DBA when certain events occur or when specific counters reach levels that might produce performance problems. Such a capability is critical for effective online/realtime event monitoring.

DB2 PE for Multiplatforms is expected to provide similar functionality in future.

Depending on whether the application workload is OLTP, Business Intelligence or mixed, the DBA needs to create and monitor custom data views for his environment. Note that the DBA may need to monitor different data views during the day, depending upon the changing application workload.

In the following sections, we describe a few data views that monitor potential performance problems in an OLTP and Business Intelligence workload environment. We categorized data views for our Trade 2 application as belonging to three groups, as follows:

- COMMON data views that apply to both OLTP and Business Intelligence workloads such as hit ratios of data, indexes and specific tablespaces.
- DW (data warehouse) or Business Intelligence data views such as sort overflows percentages, catalog/package cache overflows and hash join data exceeded available sort heap space.
- OLTP data views such as lock waits, deadlocks detected, lock time-outs and lock escalations.

Figure 4-27 shows the various data views defined in each of these groups for our target environment, with a graphical view of the buffer pool hit ratio data.

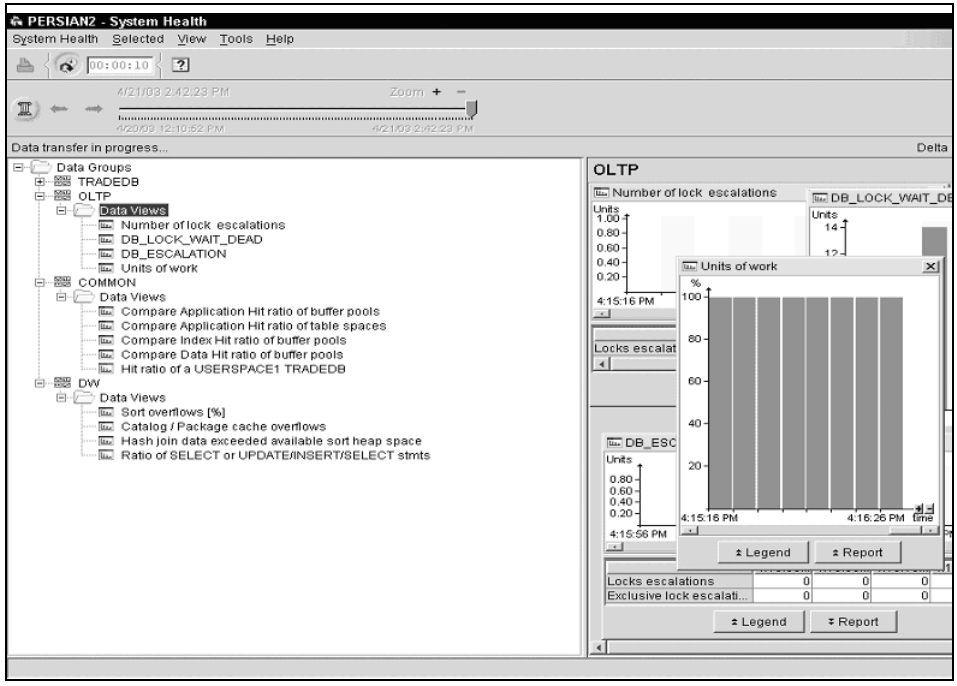


Figure 4-27 typical data views for online/realtime event monitoring

Figure 4-28 shows data views with hit ratios for specific tablespaces and buffer pools.

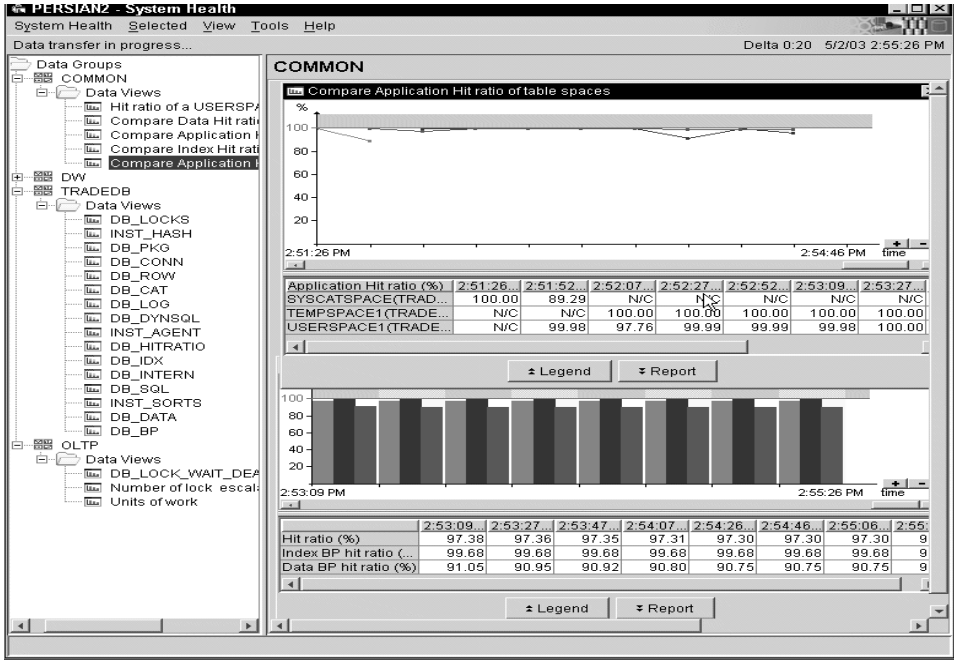


Figure 4-28 System Health - table space and buffer pool hit ratio

The following sections cover two online/realtime event monitoring scenarios:

- Lock information data view, which is more applicable to the OLTP environment.
- Sort overflow (%) data view, which is more applicable to the Business Intelligence environment.

In both the data views, we chose not to filter out any data using the threshold capability, since our contrived workloads did not provide us with sufficient scope for filtering counter values.

The application, environment configuration, and monitor level settings were identical to the ones described in 4.2.1, “Lock waits due to the default LOCKTIMEOUT parameter value” on page 66. The workloads were slightly different for each of the two scenarios, as will be described later. The triggering event in the case of both scenarios was the DBA looking at the data views for potential problems and making judgements on whether or not to take follow up action.

4.3.1 Lock information data view

The workload used for this scenario was identical to the one used in “Workload used” on page 69, with the LOCKTIMEOUT parameter set to 10 seconds and the LOCKLIST being undersized as described in 4.2.2, “Lock waits, deadlocks and time-outs due to lock escalations” on page 79.

Figure 4-29 shows the System Health window with two data views for the TRADEDB database, of which our focus is on DB_LOCKS. We used an auto refresh cycle of 20 seconds. The DB_LOCKS data view displays counter values of locks waits, deadlocks detected, locks escalations, exclusive lock escalations, and locks time-outs.

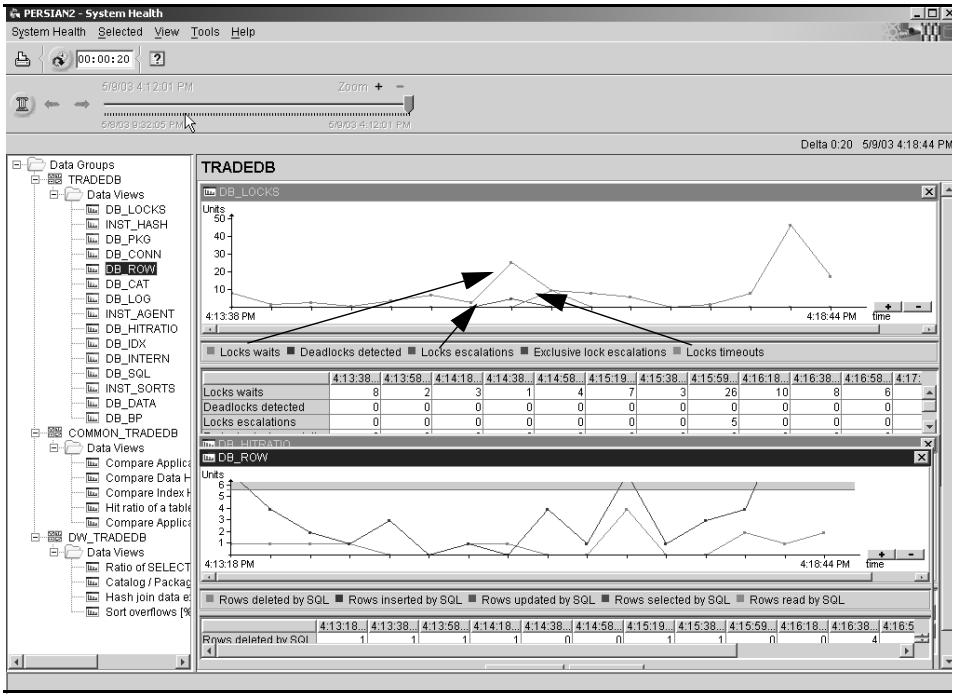


Figure 4-29 System Health - lock information, etc.

As mentioned earlier, the DBA has to constantly monitor the data views, looking for potential trends that may need problem diagnosis and resolution, since no alerts are associated with data views. The triggering event is therefore entirely DBA-driven.

The DBA may view the lock waits and lock escalations displayed in the DB_LOCKS data view in Figure 4-29, and make a judgement about whether or not to take action.

Should the DBA choose to investigate the problem, then the steps for problem diagnosis would be identical to the ones described in the scenario 4.2.2, “Lock waits, deadlocks and time-outs due to lock escalations” on page 79.

4.3.2 Sort overflow (%) data view

DB2 uses sorts to perform joins, remove duplicates and return results in a user-desired sequence. Sorts are performed in a sort heap, and rows to be sorted overflow to disk when there is insufficient sort heap memory. Sorts requiring sort overflows perform less efficiently than those that complete in sort heap memory.

The presence of a large number of sorts and the occurrence of sort overflows may indicate a performance problem requiring tuning of either the sort heap (SORTHEAP database configuration parameter) and/or the SQL application. Therefore, the number of sorts and sort overflows should be minimized. This can be achieved by encouraging index use in the access plans and thereby eliminating sorts, or increasing the sort heap size.

Refer to the *IBM DB2 UDB Administration Guide: Performance*, SC09-4821 for a detailed discussion of sort performance considerations.

To generate information of interest for the data view, we generated three queries against tables in the TRADEDB database, and ran them concurrently.

Figure 4-30 on page 98 shows the System Health window with three data views for the TRADEDB database, of which our focus is on the Sort overflows (%). We used an auto refresh cycle of 20 seconds. The Sort overflows (%) data view displays counter values of total sorts and sort overflows (%).

Note: In our contrived workload, we made the sort overflows repeatable so that we could go through the steps of problem diagnosis. In the real world, where such events may not be repeatable, the DBA may have to resort to other means, such as setting up a regression test environment to perform problem diagnosis.

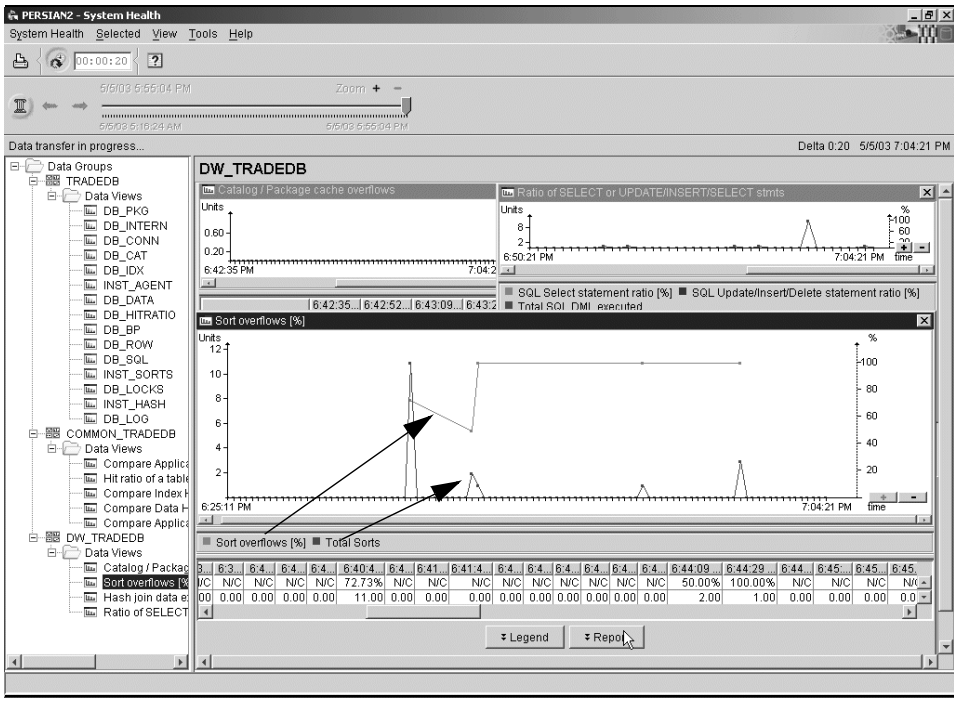


Figure 4-30 System Health - Sort overflows (%), etc.

For the purposes of this scenario, we concluded from the data view that the sort overflows were unacceptable and needed to be diagnosed and resolved. Given the repeatability of our scenario, we used the event monitor functionality of DB2 PE's Performance Warehouse to generate an SQL Activity Report for the duration of the execution of these queries. The mechanics of producing the SQL Activity Report are not shown here, but the general steps involved are described in "Performance Warehouse" on page 35. The report highlighted three queries that triggered sort overflows. Each of these queries is highlighted in Figure 4-31 on page 99, Figure 4-32 on page 99 and Figure 4-33 on page 100, respectively. The Sort overflows column in each of these figures has a non-zero value, indicating the triggering of a sort overflow.

The next steps were to:

1. Determine the access plan of each of the queries
2. Use Index Advisor for recommended indexes for each query
3. Apply best practices

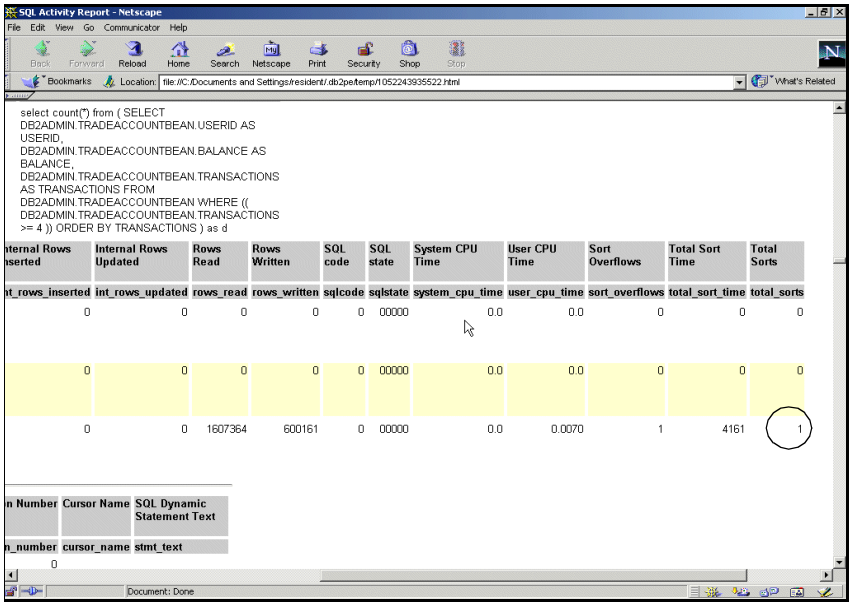


Figure 4-31 SQL Activity Report - Identify Sort Overflow - Query D

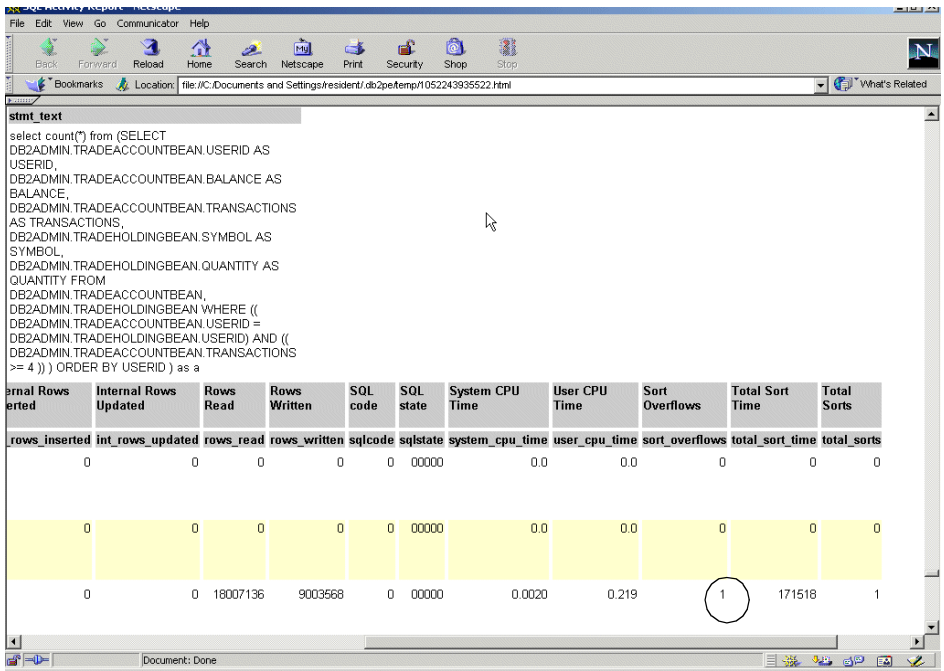


Figure 4-32 SQL Activity Report - Identify Sort Overflow - Query A

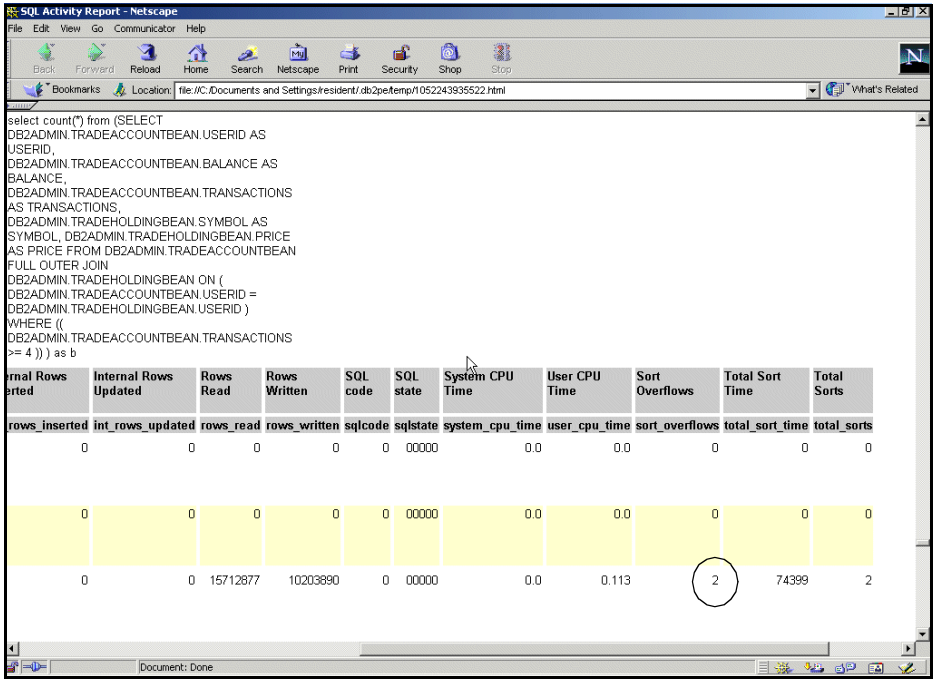


Figure 4-33 SQL Activity Report - Identify Sort Overflow - Query B

Determining the access plan

We used the Command Center to view the access plan for each of the queries, as follows.

Query A

```
select count(*)
  from (select DB2ADMIN.TRADEACCOUNTBEAN.USERID as USERID,
               DB2ADMIN.TRADEACCOUNTBEAN.BALANCE as BALANCE,
               DB2ADMIN.TRADEACCOUNTBEAN.TRANSACTIONS as TRANSACTIONS,
               DB2ADMIN.TRADEHOLDINGBEAN.SYMBOL as SYMBOL,
               DB2ADMIN.TRADEHOLDINGBEAN.QUANTITY as QUANTITY
          from DB2ADMIN.TRADEACCOUNTBEAN, DB2ADMIN.TRADEHOLDINGBEAN
          where ((DB2ADMIN.TRADEACCOUNTBEAN.USERID =
                  DB2ADMIN.TRADEHOLDINGBEAN.USERID)
                 and (( DB2ADMIN.TRADEACCOUNTBEAN.TRANSACTIONS >= 4 )) )
          order by USERID
        ) as a;
```

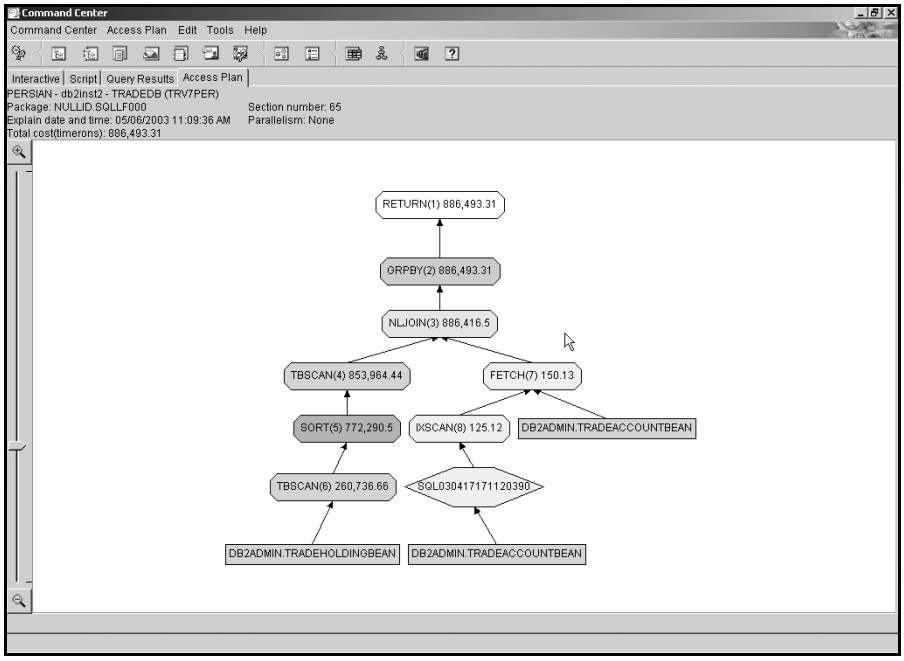


Figure 4-34 Command Center - Access Plan - Query A

Figure 4-34 shows index access to the TRADEACCOUNTBEAN table, while a table scan is used to access the TRADEHOLDINGBEAN table.

Query B

This query differs from Query A in that it uses a full outer join as compared to an inner join in Query A.

```
select count(*)
  from (select DB2ADMIN.TRADEACCOUNTBEAN.USERID as USERID,
               DB2ADMIN.TRADEACCOUNTBEAN.BALANCE as BALANCE,
               DB2ADMIN.TRADEACCOUNTBEAN.TRANSACTIONS as TRANSACTIONS,
               DB2ADMIN.TRADEHOLDINGBEAN.SYMBOL as SYMBOL,
               DB2ADMIN.TRADEHOLDINGBEAN.PRICE as PRICE
          from DB2ADMIN.TRADEACCOUNTBEAN
          full outer join DB2ADMIN.TRADEHOLDINGBEAN
            on (DB2ADMIN.TRADEACCOUNTBEAN.USERID =
                DB2ADMIN.TRADEHOLDINGBEAN.USERID )
         where (( DB2ADMIN.TRADEACCOUNTBEAN.TRANSACTIONS >= 4 ))
       ) as b
;
```

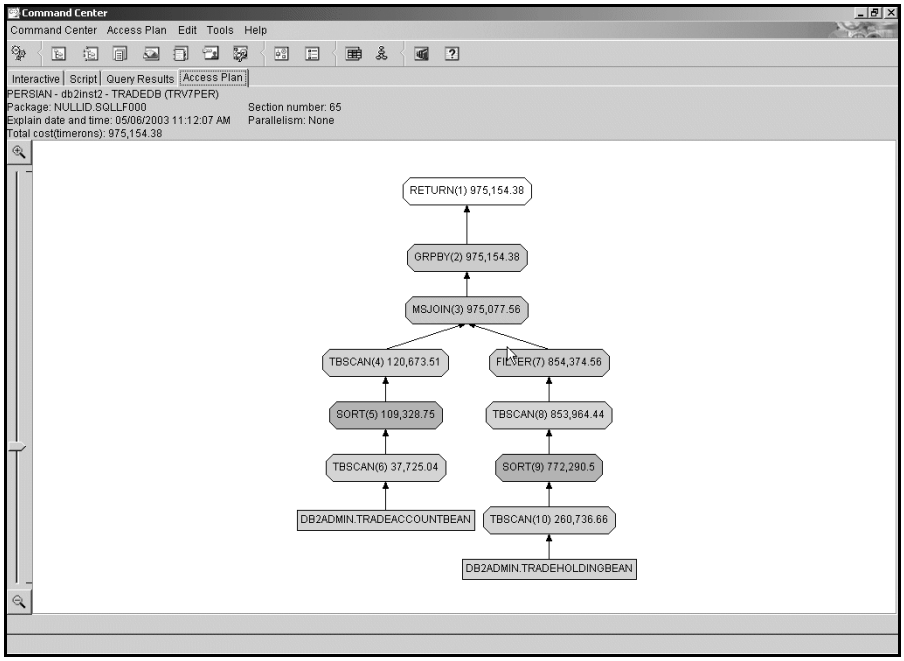


Figure 4-35 Command Center - Access Plan - Query B

Figure 4-35 shows no index access to the TRADEACCOUNTBEAN table, while continuing to use a table scan allows access to the TRADEHOLDINGBEAN table.This time, a merge join (MSJOIN) is used instead of the nested loop join used with Query A, because of the full outer join.

Query D

Query D is a much simpler query that does not involve joins. However, since the result data is to be presented in transactions sequence, a sort appears necessary.

```
select count(*) from (
   select DB2ADMIN.TRADEACCOUNTBEAN.USERID as USERID,
       DB2ADMIN.TRADEACCOUNTBEAN.BALANCE as BALANCE,
       DB2ADMIN.TRADEACCOUNTBEAN.TRANSACTIONS as TRANSACTIONS
   from DB2ADMIN.TRADEACCOUNTBEAN
   where (( DB2ADMIN.TRADEACCOUNTBEAN.TRANSACTIONS >= 4 ))
order by TRANSACTIONS
) as d
;
```

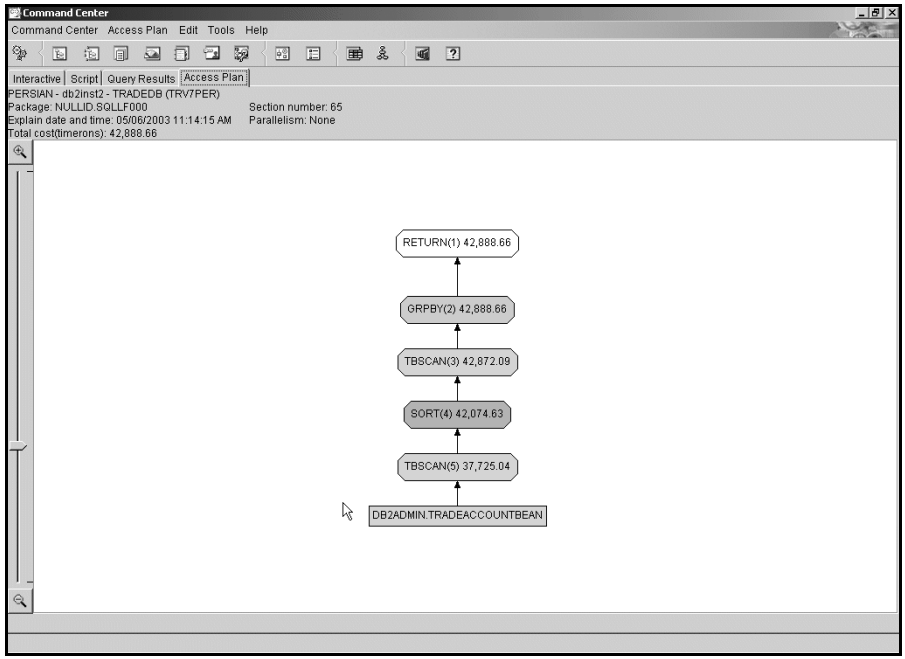


Figure 4-36 Command Center - Access Plan - Query D

Figure 4-36 shows no index access to the TRADEACCOUNTBEAN table.

Using Index Advisor for recommended indexes

We ran the Index Advisor (**`db2advis`**) on each of these three queries for recommendations on creating indexes on the target tables, and the potential performance gains achievable.

Note: It is assumed in this discussion that catalog statistics for the tables involved in the queries are current.

Query A

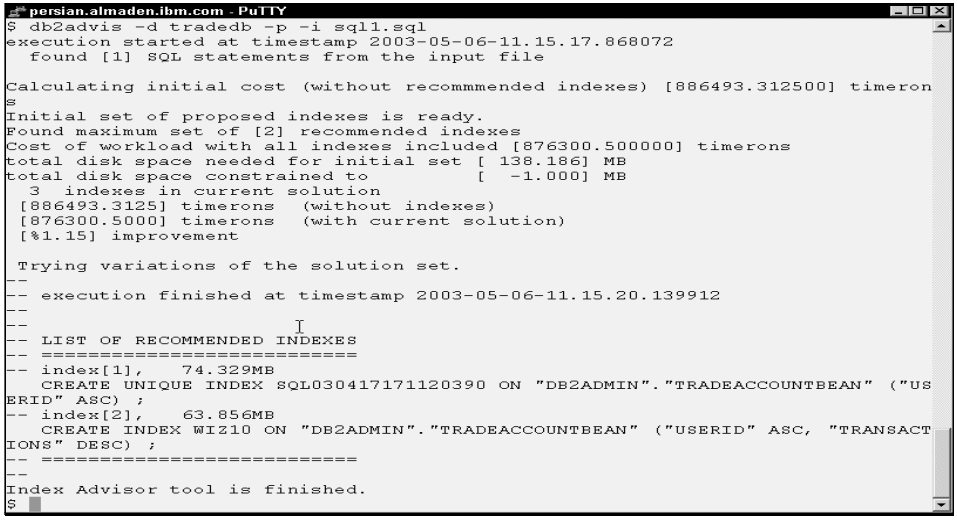


Figure 4-37 Index Advisor - Query A

Figure 4-37 recommends two indexes that will consume 130 MB of disk space, with a projected performance gain of only 1.15%; the performance gain projected is not worth the cost of creating the indexes.

Important: The performance gains projected may not reflect the actual gains achievable. One needs to actually create the indexes and run the query to determine the performance gains obtained, if any. For the purposes of our discussion, however, we chose to assume that the information provided by the Index Advisor was accurate, an assumption that should not be made in real world environments.

Additionally, the creation of indexes will have a negative impact on the performance of transactions performing inserts/updates/deletes, as well as the performance of certain utilities. We have ignored these considerations in our controlled environment, an assumption that should not be made in real world environments.

Query B

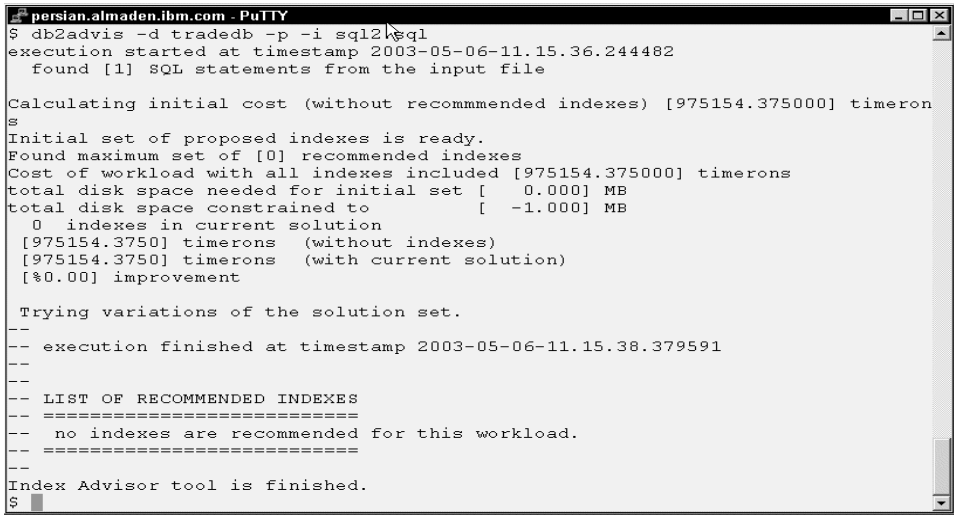


Figure 4-38 Index Advisor - Query B

Figure 4-38 recommends no index for this query.

Query D

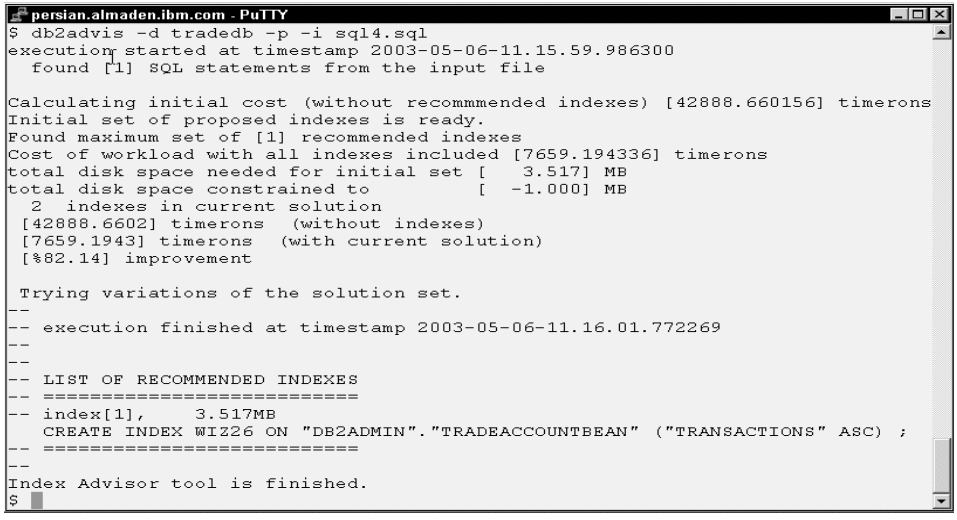


Figure 4-39 Index Advisor - Query D

Figure 4-39 on page 105 recommends the creation of a single index that will consume only 3.5 MB of disk space, while projecting an improvement of be more than 80%, a cost-effective recommendation.

Application of best practices

Creating an index for Query D seems beneficial. The fact that indexes did not seem appropriate for Query A and Query B meant that we should consider increasing the sort heap size to avoid sort overflows.

Refer to the *IBM DB2 UDB Administration Guide: Performance*, SC09-4821 for a detailed discussion of choosing the sort heap size.

4.4 Routine monitoring considerations

As discussed in 1.2.1, “Routine monitoring” on page 4, routine monitoring involves collecting information about the workload and the stress on the system during periods of normal and peak activity, for capacity planning purposes as well as to identify potential problems down the road. A critical requirement of routine monitoring is to incur minimal overhead on the system being monitored, given the continuous nature of such monitoring.

Routine monitoring typically involves ascertaining minimums, maximums, average, hit ratios, etc. for key performance elements such as heaps, transactions, users, CPU, memory, locks, disk space, and SQL over days and weeks. Such reporting requires the trace data to be stored in a repository such as the DB2 PE Performance Warehouse for subsequent analysis and reporting.

Important: While the DB2 PE Performance Warehouse is capable of storing trace data for analysis and reporting, the source of this trace data is currently limited to Event Monitor data only. Since Event Monitor tends to incur high overhead, it should normally only be used in short bursts for problem determination.

However, you may choose to use Event Monitor data for routine monitoring if the overhead incurred is acceptable for your workload in your particular environment.

DB2 PE does provide a history mode for collecting instance wide data using snapshots, as described in “History mode” on page 21. Currently, this data is stored in wraparound data sets that can only be viewed online, with no capability to provide long-term trend analysis reports.

A

Sample applications

This appendix briefly describes the Trade2 application, the trade database tables, the SQL queries used in the various workloads, and akstress control information.

A.1 Trade 2 application

The Trade 2 benchmark, also called the WebSphere Performance Benchmark Sample, was developed by IBM and is publicly available. This application models an online brokerage firm providing Web-based services such as login, buying, selling, getting quotes and more. Figure A-1 shows the various application components and model-view-controller topology.

The Trade 2 application is a collection of Java™ classes, Java Servlets, Java Server Pages and Enterprise Java Beans (EJBs) that service requests made by registered users. This application runs as a single Java process which is managed by WebSphere Application Server.

This workload exercises the entire solution stack that consists of the WebSphere Application Server, JVM, and the Just-In-Time (JIT) compiler, the HTTP server, the DB2 Database Server and the DB2 client, the AIX operating system, and the system hardware.

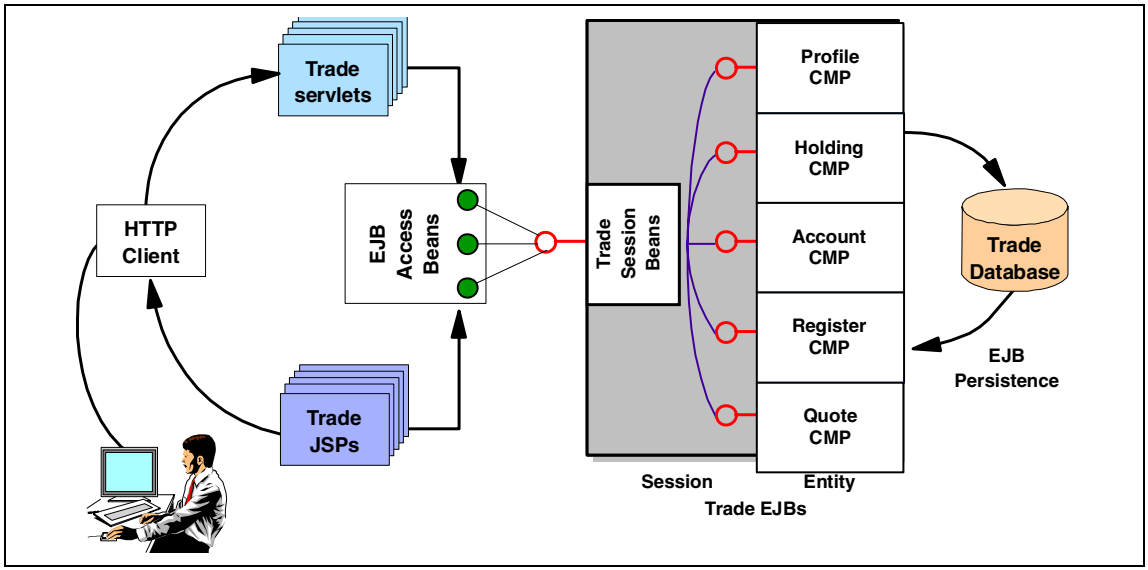


Figure A-1 Trade 2 application

Further details about this application including sample code can be obtained from:

http://www-3.ibm.com/software/webservers/appserv/wpbs_download.html

A.2 Trade 2 database tables

Figure A-2 describes the tables defined in the Trade database, and the relationships between them.

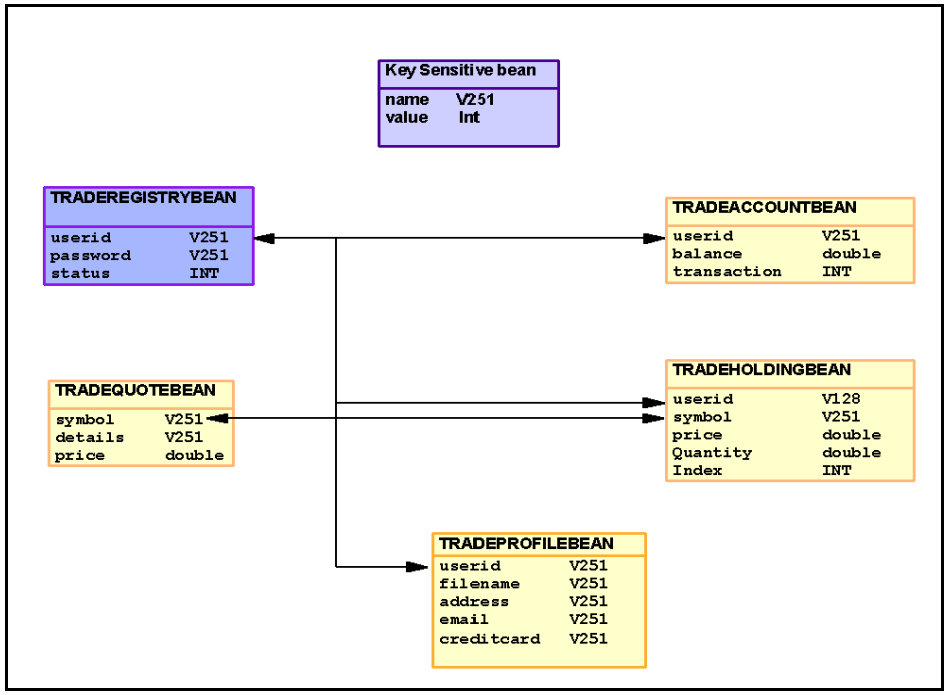


Figure A-2 TradeDB-Datamodel

To simulate different situations, we used the following SQL statements. To create locks, we used the DB2 command line processor with option **+c**, meaning without commit (no internal commit or rollback).

A.2.1 Select statements

Query A

```
SELECT DB2ADMIN.TRADEACCOUNTBEAN.USERID AS
USERID,DB2ADMIN.TRADEACCOUNTBEAN.BALANCE AS BALANCE,
DB2ADMIN.TRADEACCOUNTBEAN.TRANSACTIONS AS TRANSACTIONS,
DB2ADMIN.TRADEHOLDINGBEAN.SYMBOL AS SYMBOL,
DB2ADMIN.TRADEHOLDINGBEAN.QUANTITY AS QUANTITY FROM
DB2ADMIN.TRADEACCOUNTBEAN, DB2ADMIN.TRADEHOLDINGBEAN WHERE ((
DB2ADMIN.TRADEACCOUNTBEAN.USERID = DB2ADMIN.TRADEHOLDINGBEAN.USERID) AND
(( DB2ADMIN.TRADEACCOUNTBEAN.TRANSACTIONS >= 4 )) ) ORDER BY USERID
```

Query B

```
SELECT DB2ADMIN.TRADEACCOUNTBEAN.USERID AS
USERID,DB2ADMIN.TRADEACCOUNTBEAN.BALANCE AS BALANCE,
       DB2ADMIN.TRADEACCOUNTBEAN.TRANSACTIONS AS TRANSACTIONS,
DB2ADMIN.TRADEHOLDINGBEAN.SYMBOL AS SYMBOL,
       DB2ADMIN.TRADEHOLDINGBEAN.PRICE AS PRICE
  FROM DB2ADMIN.TRADEACCOUNTBEAN
  FULL OUTER JOIN DB2ADMIN.TRADEHOLDINGBEAN
    ON ( DB2ADMIN.TRADEACCOUNTBEAN.USERID =
DB2ADMIN.TRADEHOLDINGBEAN.USERID )
 WHERE (( DB2ADMIN.TRADEACCOUNTBEAN.TRANSACTIONS >= 4 ))
```

Query D

```
SELECT DB2ADMIN.TRADEACCOUNTBEAN.USERID AS USERID,
DB2ADMIN.TRADEACCOUNTBEAN.BALANCE AS BALANCE,
DB2ADMIN.TRADEACCOUNTBEAN.TRANSACTIONS AS TRANSACTIONS FROM
DB2ADMIN.TRADEACCOUNTBEAN WHERE (DB2ADMIN.TRADEACCOUNTBEAN.TRANSACTIONS >=
4 )
ORDER BY TRANSACTIONS
```

A.2.2 Update statements

SQL for creating lock waits

```
db2 +c UPDATE DB2ADMIN.TRADEQUOTEBEAN SET DETAILS = 'testing for Lock
Waits....' WHERE (( DB2ADMIN.TRADEQUOTEBEAN.PRICE = 100.00 ))

db2 +c UPDATE DB2ADMIN.TRADEREGISTRYBEAN SET STATUS = -1 WHERE USERID like
'ru:%'
```

```
db2 +c UPDATE DB2ADMIN.TRADEQUOTEBEAN SET DETAILS = 'testing for Lock
Escalations' WHERE (( DB2ADMIN.TRADEQUOTEBEAN.PRICE >= 100.00 ))

db2 +c UPDATE DB2ADMIN.TRADEHOLDINGBEAN SET SYMBOL = 'Testing' WHERE ((
DB2ADMIN.TRADEHOLDINGBEAN.INDX <> 7 ))
```

A.3 Akstress

Web Performance Tools, or WPT (formerly AKtools), is a set of applications allowing a user to test a Web server, a Web site, and/or a Web application.

Note: This product is available on the IBM alphaworks Web site at:

`http://www.alphaworks.ibm.com/tech/wptools`

Version 1.9 of WPT consists of two applications:

- **`akstress`** is a high-performance, simple, threaded HTTP engine which is capable of simulating hundreds or even thousands of HTTP clients, using a highly configurable set of directives in a human readable and easily modified configuration file.
- **`akrecord`** is a simple eavesdropping proxy that will record a user's session against a Web server for later playback in **`akstress`**.

When the two applications are combined, it becomes very easy to quickly build an **`akstress`** configuration, which, with minor tuning, allows a user to evaluate the usability of a server, site, or Web application.

`akstress` is built on the code from several other IBM stress test tools. Those tools have been used for the last several years for things like HTTP/1.1 verification testing, large Web site stress analysis, HTTP Server SVT testing, and Web server unit testing efforts.

The following is a list of some of the available functions and advantages of this tool:

- Fully configurable HTTP headers
- SSL support
- Support for HTTP/1.1 functions, including persistent connections and chunked-transfer encoding
- Built-in cookie cache (for session testing)
- Result verification
- Full logging

- Overall and request-level statistics
- Simplicity of use, no requirement for third party interpreters, etc.
- Socks support for recording and replay

Important: It is important to note that WPT is *not* a replacement for some of the high-end Web stress tools. It was created to be either a “quick and dirty” testing tool, or used in environments where the purchase of high-end tools is prohibitive.

We created a separate batch and acf file for each database.One example of each file follows.

A.3.1 Batch

The following batch file is use to execute **akstress** with the right acf file.

```
start ..\AK1910\bin\akstress -config PERSIAN2_mmtrade2scenario.acf   -host
boron
```

A.3.2 ACF-File PERSIAN2_mmtrade2scenario.acf

The acf file is a datafile with control information for **akstress**, and includes details about the Java servlet to be invoked and the number of concurrent users to simulate.

```
#
# Global Setting information
# Threads - the number of threads to run against the target server
# TotalPageRequests - the number of page requests to make
#                   - can consist of multiple requests
# ErrorLog - the file that errors will be logged to
# StatServerPort - the port that the stat server will listen for
# RandomEarlyClosePercentage - percentage of requests that will be closed
early
#

global_settings
   Clients                              200
   ThreadsPerClient                     2
   TotalPageRequests 200000
   ErrorLog log\PERSIAN2_MMerror.log
   ThinkMinimum1
   ThinkMaximum2
   CookieCache  on
   PurgeCookieCache off
   CookieIgnoreDefaultExpires on
```

```
   RampUp
   TimedRun 1800            #seconds

end_global_settings

stats_definition
   RequestStatistics on
   PageStatistics    on
   PageLog           log\PERSIAN2_MMpage_V8P.log
   RequestLog        log\PERSIAN2_MMrequest_V8P.log
   SummaryLog        log\PERSIAN2_MMsummary_V8P.log
end_stats_definition

header_definition header0
#  Connection: Keep-Alive
   User-Agent: Mozilla/4.72 [en] (Windows NT 5.0; U)
   Host: <<HOST>>
   Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
   Accept-Encoding: gzip
   Accept-Language: en,pdf
   Accept-Charset: iso-8859-1,*,utf-8
end_header_definition

request_definition request0
   Header header0
   Method GET
   URI /WebSphereSamples/TradeSampleV7P/servlet/TradeScenarioServlet
   Protocol HTTP/1.0
   ResultCode 200
end_request_definition


start_page page0
   Request request0
end_page_definition
```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 116.

- *DB2 Performance Expert for z/OS*, SG24-6867
- *DB2 UDB V7.1 Performance Tuning Guide,* SG24-6012
- *DB2 UDB Exploitation of the Windows Environment,* SG24-6893
- *Database Performance Tuning on AIX,* SG24-5511
- *DB2 UDB/WebSphere Performance Tuning Guide,* SG24-6417

Other resources

These publications are also relevant as further information sources:

- *IBM DB2 Performance Expert: Installation and Customization for z/OS and Multiplatforms*, SC27-1646-04
- *IBM DB2 Performance Expert: Monitoring Performance from the Workstation for z/OS and Multiplatforms*, SC27-1645-04

Version 7.x:

- *DB2 UDB Administration Guide: Planning*, SC09-2946
- *DB2 UDB Administration Guide: Implementation*, SC09-2944
- *DB2 UDB Administration Guide: Performance*, SC09-2945
- *DB2 UDB System Monitor Guide and Reference,* SC09-2956

Version 8:

- *DB2 UDB Administration Guide: Planning*, SC09-4822
- *DB2 UDB Administration Guide: Implementation*, SC09-4820
- *DB2 UDB Administration Guide: Performance*, SC09-4821
- *DB2 UDB System Monitor Guide and Reference,* SC09-4847

Referenced Web sites

These Web sites are also relevant as further information sources:

- IBM DB2 Universal Database for Linux, UNIX and Windows

  `http://www.ibm.com/software/data/db2/udb/`

- DB2 Performance Expert

  `http://www.ibm.com/software/data/db2imstools/db2tools/db2pe/index.html`

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the **CD-ROMs** button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

W

IBM Redbooks

DB2 UDB Performance Expert for Multiplatforms: A Usage Guide

(0.2”spine)
0.17”<->0.473”
90<->249 pages

DB2 UDB Performance Expert for Multiplatforms

A Usage Guide

DB2 Performance Expert overview

DB2 Performance Expert installation and customization

Usage scenarios

This IBM Redbook provides guidelines for using the new DB2 UDB Performance Expert tool for Multiplatforms, and is aimed at a target audience of DB2 database administrators (DBAs).

We provide an overview of the architecture of DB2 Performance Expert for Multiplatforms and describe its main components and features.

We highlight some of the key considerations in configuring DB2 Performance Expert for Multiplatforms in your environment and provide recommendations for optimal use.

Finally, we discuss some of the commonly encountered problems faced by a DBA when managing a DB2 environment, and describe how the tool can be used to diagnose and resolve these performance problems.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6436-00 ISBN 0738428248