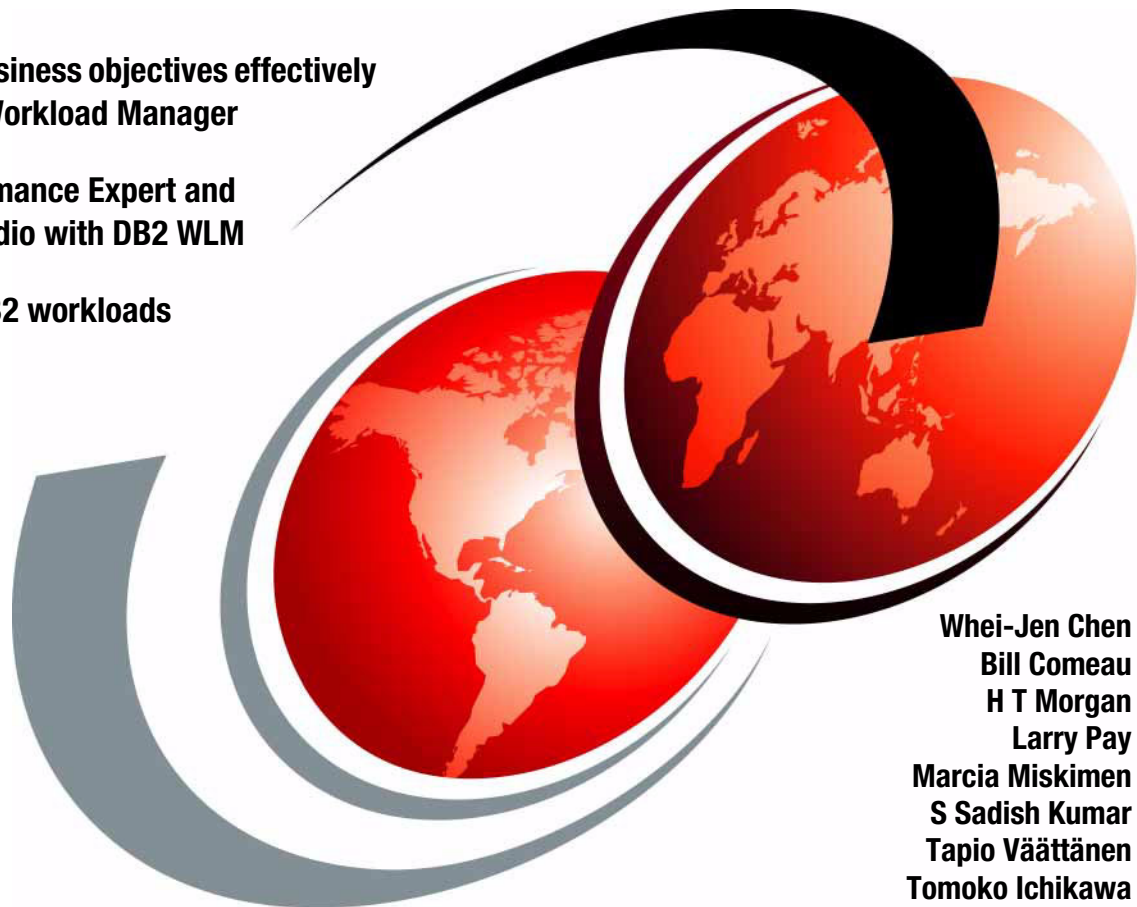


DB2 Workload Manager for Linux, UNIX, and Windows

Achieve business objectives effectively
with DB2 Workload Manager

Use Performance Expert and
Design Studio with DB2 WLM

Manage DB2 workloads
proactively



Whei-Jen Chen
Bill Comeau
H T Morgan
Larry Pay
Marcia Miskimen
S Sadish Kumar
Tapio Väättänen
Tomoko Ichikawa



International Technical Support Organization

**DB2 Workload Manager for Linux, UNIX, and
Windows**

November 2007

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

First Edition (November 2007)

This edition applies to DB2 9.5 for Linux, UNIX, and Windows.

This document created or updated on October 2, 2007.

© Copyright International Business Machines Corporation 2007. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks	x
Preface	xi
The team that wrote this book	xi
Acknowledgements	xiii
Become a published author	xiv
Comments welcome	xiv
Chapter 1. Introduction	1
1.1 Workload management	2
1.2 DB2 Workload Manager	3
Chapter 2. WLM architecture and features	7
2.1 DB2 Workload Manager concepts	8
2.1.1 DB2 service classes	9
2.1.2 DB2 workloads	11
2.1.3 DB2 thresholds	14
2.1.4 Work class sets and work action sets	17
2.2 Architecture	20
2.3 DB2 WLM monitor and control capabilities	24
2.3.1 Real-time monitoring	24
2.3.2 Statistics table functions	25
2.3.3 Event monitors for DB2 WLM	26
2.3.4 WLM stored procedures	29
2.4 New database configuration parameter and catalog tables	29
2.5 Working with WLM SQL and objects	30
2.5.1 DB2 Service classes	31
2.5.2 DB2 Workloads	32
2.5.3 DB2 Thresholds	34
2.5.4 DB2 work classes	36
2.5.5 DB2 work action set	37
Chapter 3. Getting started	41
3.1 System requirements	42
3.1.1 Hardware	42
3.1.2 Software	42
3.1.3 Platforms supported	43

3.2	Planning DB2 environment	44
3.3	Lab environment	45
3.3.1	Lab systems	45
3.3.2	AIX server configuration	46
3.3.3	TPC-H	48
3.4	Installing DB2	49
3.5	First steps	53
3.5.1	The default DB2 WLM configuration	53
3.5.2	Monitoring the default WLM environment	55
	Chapter 4. Customizing the WLM execution environments	61
4.1	Stages of workload management	62
4.2	Identify the work	63
4.2.1	Workload identify worksheet	65
4.3	Manage the work	66
4.3.1	Creating the service classes	67
4.3.2	Creating the workloads	68
4.3.3	Allowing use of the WLM setup	70
4.3.4	Creating the event monitor	70
4.3.5	Using SYSDEFAULTADMWORKLOAD	71
4.4	Monitor the work	73
4.5	Summary	79
	Chapter 5. Monitoring	83
5.1	Real-time monitoring	84
5.1.1	Workload management table functions	84
5.1.2	Workload management stored procedures	95
5.1.3	db2pd command for workload management	97
5.2	Historical monitoring	99
5.2.1	Activities event monitor	100
5.2.2	Threshold violations event monitor	109
5.2.3	Statistics event monitor	113
5.3	Workload profiling and capturing	126
5.3.1	Monitoring overall database system behavior	127
5.3.2	Monitoring the queued job	142
5.3.3	Identifying query with long runtime	147
	Chapter 6. WLM Sample Scenario - OLTP	151
6.1	Business objectives	152
6.2	Identification	152
6.3	Consistent response time	153
6.3.1	Define DB2 workloads and service classes	154
6.3.2	Monitoring	156
6.3.3	Summary	163

6.4 Mitigate long-run queries	163
6.4.1 Define DB2 workloads and service classes	164
6.4.2 Define controls	164
6.4.3 Monitoring	165
6.4.4 Summary	166
6.5 Prevent concurrent queries hogging the system	167
6.5.1 Identification	167
6.5.2 Define work classes for control	167
6.5.3 Monitoring	168
6.5.4 Summary	170
6.6 Stop user connections idle for more than 30 minutes	170
6.6.1 Identification	170
6.6.2 Define controls	170
6.6.3 Monitoring	171
6.6.4 Summary	172
Chapter 7. WLM sample scenarios - Mixed OLTP and DSS environment	173
7.1 Business objectives	174
7.2 Identify the work	174
7.3 Manage the work	176
7.3.1 Enabling the instance user ID to alter AIX priorities	176
7.3.2 Creating the service classes definitions	177
7.3.3 Creating the workload definitions	179
7.3.4 Finalizing the setup	179
7.4 Monitoring the work	180
7.4.1 Checking the agent priorities and prefetchers	180
7.4.2 Monitoring and analyzing the service classes	185
7.5 Summary	191
Chapter 8. AIX Workload Manager considerations	193
8.1 AIX WLM overview	194
8.1.1 Service classes	194
8.1.2 Monitoring	197
8.1.3 Configuring AIX WLM	198
8.2 Using DB2 WLM and AIX WLM	202
8.2.1 General guidelines	202
8.2.2 Mapping schemes	203
8.2.3 Integrating DB2 service classes with AIX service classes	205
8.2.4 Monitoring	214
Chapter 9. WLM sample scenarios - other usage	221
9.1 Capacity planning	222
9.1.1 The workload environment	223
9.1.2 Collecting the trending data	223

9.1.3	Monitoring and analysis	224
9.1.4	Summary	228
9.2	Chargeback accounting	229
9.2.1	Business objectives	230
9.2.2	Defining workload profile	230
9.2.3	Monitoring	230
9.2.4	Summary	232
Chapter 10.	DB2 WLM and DWE Design Studio	235
10.1	DB2 Warehouse Design Studio overview	236
10.1.1	Workload management support	237
10.1.2	Installing DB2 DWE Design Studio	238
10.2	Getting start	238
10.2.1	Workload Management Scheme	242
10.3	Managing database workloads using Design Studio	245
10.3.1	Create workload scheme by objective	247
10.3.2	Create workload scheme by yourself	288
10.3.3	Create workload scheme by reverse engineering	292
10.4	Execute a workload management scheme	296
10.5	AIX WLM management	303
10.5.1	Creating operating system service classes and limits	303
10.5.2	Configure AIX WLM using Design Studio	311
Chapter 11.	DB2 Workload Manager and DB2 Performance Expert	319
11.1	DB2 Performance Expert overview	320
11.2	Monitoring your DB2 environment	321
11.2.1	Monitoring instance and database statistics	326
11.3	Monitoring DB2 Workload Manager	329
11.3.1	Workload Management Key Performance Indicators	329
11.3.2	Viewing workload management definitions	331
11.3.3	Viewing Workload Management statistics	334
11.4	DB2 Performance Expert technical information	350
Chapter 12.	Administration	359
12.1	WLM logs and maintenance	360
12.2	WLM problem diagnosis	363
12.3	WLM backup and recovery	367
12.4	WLM authorization	368
Chapter 13.	Query Patroller and DB2 Governor	371
13.1	Query Patroller and DB2 Governor background	372
13.1.1	Query Patroller	372
13.1.2	DB2 Governor	373
13.1.3	Differences between QP, Governor, and WLM	374

13.2 Co-existing	375
13.3 Transitioning from Query Patroller and Governor	376
13.3.1 Is there a migration tool?	376
13.3.2 Re-examining goals	377
13.3.3 Considerations when migrating from a QP environment.	377
13.3.4 Considerations when migrating from a DB2 Governor environment. .	383
13.3.5 Historical information in QP control tables	384
Related publications	389
IBM Redbooks	389
Other publications	389
Online resources	390
How to get Redbooks	391
Help from IBM	391
Index	393

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 illustrate 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.

Trademarks

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

Redbooks (logo) ®
eServer™
pSeries®
xSeries®
AIX®

DB2®
IBM®
OpenPower™
POWER™
Redbooks®

System i™
System p™
System p5™
WebSphere®

The following terms are trademarks of other companies:

Snapshot, and the Network Appliance logo are trademarks or registered trademarks of Network Appliance, Inc. in the U.S. and other countries.

AMD, AMD Athlon, the AMD Arrow logo, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

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

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

Intel, Itanium, Pentium, Xeon, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

DB2® Workload Manager (WLM) introduces a significant evolution in the capabilities available to database administrators for controlling and monitoring executing work within DB2. This new WLM technology is directly incorporated into the DB2 engine infrastructure to allow handling the higher volumes with minimal overhead. It is also enabled for tighter integration with external workload management products such as provided by AIX® WLM.

This book discusses the features and functions of DB2 Workload Manager for Linux, UNIX®, and Windows®. We describe DB2 WLM architecture, components, and the WLM specific SQL statements. We also discuss installation, WLM methodology for customizing the DB2 WLM environment, new workload monitoring table functions, event monitors, and stored procedures. We provide examples and scenarios of using DB2 WLM to manage database activities in an OLTP, DSS, and mixed database systems. Through the use of examples, you will learn about these advanced workload management capabilities and see how they can be used to explicitly allocate CPU priority, detect and prevent “run away” queries, and to closely monitor database activity in a number of different ways.

We also discuss using Data Warehouse Edition Design Studio and DB2 Performance Expert with DB2 WLM. Finally, we give the primary differences between Workload Manager and Query Patroller as well as how they interact in DB2 9.5.

The team that wrote this book

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

Whei-Jen Chen is a Project Leader at the International Technical Support Organization, San Jose Center. She has extensive experience in application development, database design and modeling, and DB2 system administration. Whei-Jen is an IBM® Certified Solutions Expert in Database Administration and Application Development as well as an IBM Certified IT Specialist.

Bill Comeau is the technical manager for the WLM development team based in Toronto. He has been working for IBM for the last 17 years, the last seven on

workload management solutions for DB2 LUW. In addition, Bill holds an honours degree in computing science from Dalhousie University in Halifax, Nova Scotia.

H T Morgan is a Senior Software Engineer working as a Premier Support Analyst in the Information Management Software Group. As a Premier Support Analyst, he provides dedicated DB2 support for several large data warehouse customers. He has 40 years experience in the Information Technology field, with past 20 years dedicated to various aspects of DB2 development and support. Since joining IBM in 1998, he has worked in Global Services and DB2 Lab Services as a Consulting I/T Specialist; and a DB2 Premier Support Analyst. Prior to joining IBM, H.T.'s expertise in DB2 includes; application development and design, relational technology research, project management, software development management, and technical support management.

Larry Pay is a ?????title??? in ???country???. He/she has ?? years of experience in ???? field. He/she holds a degree in ???? from ????. His/her areas of expertise include ?????? He/she has written extensively on ????????

Marcia Miskimen is a Software Engineer in the USA at IBM Software Group's Silicon Valley Lab, where she is currently a specialist supporting Information Management Tools on LUW platforms. She has worked in the IT industry for over 25 years, both inside and outside of IBM. She has worked in application development, systems management, operations support, services and consulting, including 10 years as an IT Specialist in IBM Global Services. Her areas of expertise and interest include the application development life cycle, software testing, technical writing, and tools of all sorts. Marcia has co-authored several IBM Redbooks on DB2 Performance Expert and DB2 Recovery Expert.

S Sadish Kumar is a ?????title??? in ???country???. He/she has ?? years of experience in ???? field. He/she holds a degree in ???? from ????. His/her areas of expertise include ?????? He/she has written extensively on ????????

Tapio Väättänen is a ?????title??? in ???country???. He/she has ?? years of experience in ???? field. He/she holds a degree in ???? from ????. His/her areas of expertise include ?????? He/she has written extensively on ????????

Tomoko Ichikawa is an IT Specialist with IBM Systems Engineering Co., Ltd. in Japan. She has been working in DB2 technical support for four years. She has planned, developed and delivered transition workshops of DB2 UDB V8.2, V8.2.2, and V9.1 for IBMers in Japan. Her areas of expertise include application development, database performance and monitoring, and problem determination in 3-tier environment (DB2 for LUW and WebSphere® Application Server).

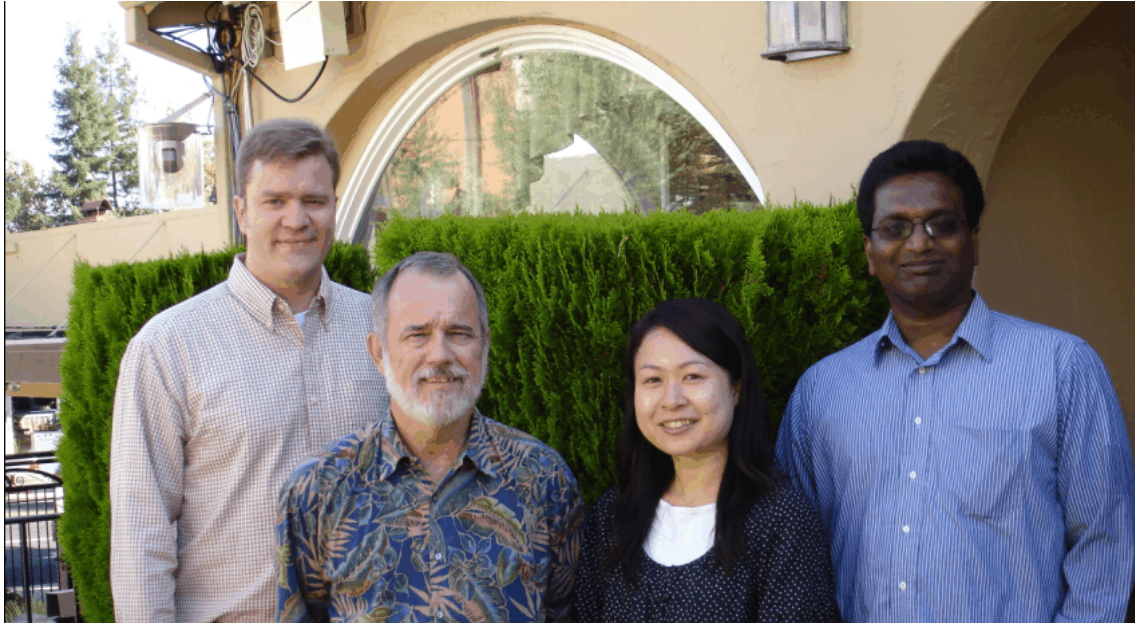


Figure 0-1 Left to right: Tapio, H T, Tomoko, and Sadish

Acknowledgements

The authors express their deep gratitude for the advice and support they received from **Paul Bird** from the IBM Toronto Laboratory.

We also thank the following people for their support and contributions to this project:

Karen Mcculloch
Louise McNicoll
Mokhtar Kandil
Francis Wong
IBM Toronto Laboratory

Keven Beck
Dinkar Rao
IBM Silicon Valley Laboratory

Tetsuya Shirai
IBM Systems Engineering, Japan

Many thanks to our support staff for their help in the preparation of this book:
Emma Jacobs, Sangam Racherla, and Deanna Polm
International Technical Support Organization, San Jose Center

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will 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 books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks® in one of the following ways:

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

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400



Introduction

In today's world, data is being collected faster than any other period in history. At the same time many businesses are trying to maximize an efficient cost model by using their hardware and software resources as heavily as possible. Data servers are being consolidated, report generation is easily accessible and users are permitted ad-hoc access to valuable data. This leads to a high potential for periodic chaos on any data server environment. Hence, we have the need for methods to return stability, predictability, and control (in the form of resource and request management and monitoring) back to data server customers in the form of workload management.

DB2 Workload Manger, integrated into the DB2 9.5 for Linux®, UNIX, and Windows, is a comprehensive workload management feature that gives you deeper insight into how your system is running, and finer control over resources and performance.

By reading this book, you will become familiar with the enhanced workload management capabilities that are included in the DB2 9.5 release.

In this chapter, we discuss the following topics:

- ▶ The background of workload management
- ▶ Concepts of workload management
- ▶ DB2 Workload Manager

1.1 Workload management

In a typical database environment, there are a wide range of database activities that can flood the data server. There are short transactional updates to a warehouse, (potentially) long reports, batch loading of data, applications calling stored procedures, and so on. These activities can come from many different sources as well, such as different users, business units, applications, multi-tier servers, and even DB2 itself.

At times (maybe most of the time), it is not uncommon to find the data server performing unexpectedly because of all the database activities. In order to keep control of the data server work, a comprehensive approach to workload management is critical.

An effective workload management environment can be broken down into four basic stages. First, and foremost, is a good understanding of the business goals you are looking to meet for this system. For example, perhaps there are updates to the database from cash registers around the country and it is critical to have a fast response time in order to keep customers from waiting. Or maybe you have a set of daily reports that have to be completed by 9 AM every morning for a review meeting.

Once the goals are established, the next stage of WLM is the identification of activities that you will be trying to manage. The wider the range of options for identification, the more likely you will be able to isolate the work you will be managing. Examples of methods of identification include by source (for example, an application or user ID that submitted a query) or by type (for example, load commands or queries that are read-only).

The third stage of WLM is the management of activities in order to meet the business goals. This would include any mechanism that can affect how an activity is executed. For example, CPU or I/O resources could be made available to a set of queries, thresholds could be introduced to limit the time spent on an activity or the number that can run concurrently, or activities could be simply stopped.

The final stage of WLM is the ability to monitor the state of activities on the data server. If you have take the time to establish business goals for the environment, then it is important to have the mechanisms in place to determine if you are meeting those goals (for example, if you have response time goals, then you need to be able to determine what the average response time is) as well as monitoring options to identify and resolve problem areas or even just getting a clear picture of what activities are running.

Monitoring information is also very useful in determining if there are modifications required to the identification or management stages. It is also a good idea to use some monitoring information before the initial configuration to validate the approach used in identification. For example, the activity information available from the monitoring should contain enough details about its source or type to help with the identification process.

Figure 1-1 illustrates these workload management stages.

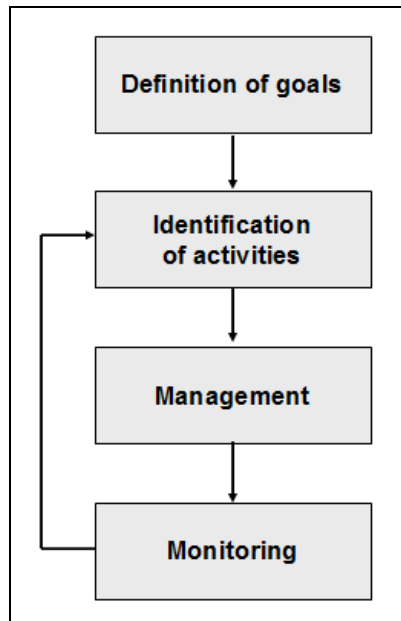


Figure 1-1 Stages of workload management

1.2 DB2 Workload Manager

The approach taken to provide a robust workload management was to first focus on establishing an execution environment. An execution environment is simply a contained area where the database activities are allowed to run, sort of a sandbox for the requests to play in.

With the execution environment concept in place, the above mentioned stages of workload management fall into place:

- ▶ Identification
In the identification stage, you focus on the assigning of database activities to an execution environment where the activities map to those established in the business goals.
- ▶ Management
In this stage, you focus on tactical controls to track and modify the execution flow of database activities in order to meet the business goals
- ▶ Monitoring
This stage provides access to the state of activities in the execution environment as well as an indication as to whether the business goals are being met.

There are a number of common database related business problems that can be addressed through an effective workload management implementation. They include:

- ▶ Protecting the system from rogue (or runaway) queries
Large resource intensive activities, like queries, can be a huge hindrance on the performance of the data server workload. Sometimes the activities are perfectly valid but, more often than not, they are simply poorly written queries or a case where too many of these expensive activities are running at the same time.
- ▶ Maintaining consistent response times for activities
It is critical to keep short transactional queries in a warehouse environment executing at a consistent and predictable pace. Often, these activities can be isolated and can easily have response times impacted by unexpected workloads.
- ▶ Protecting the data server from a system slowdown during peak periods of database activity
Normally there are periods during every day (or week, or month) where an unusually large number of database activities seem to all be executing at the same time. Not surprisingly, this often results in resource contention and slowdowns on the system.
- ▶ Explicit resource control
Resource allocation and resource contention can be a real problem on the data server machine. Customers need ways to fairly allocate resources across execution environments and limit excess resource consumption.
- ▶ Enforce Service Level Agreement (SLA) objective
Service level agreements often introduce explicit response time goals for database activities with little or no methods in place to control the workloads

and use monitor information to cheaply determine how the data server is tracking to those goals.

- ▶ Granular monitoring of database activities (ability to monitor the whole life-cycle of activities as well as the aggregate distribution)

Often a “big picture” view of data server activity is sufficient information to determine the overall health of the activity distribution. Sometimes detailed information is required for problem determination and historical analysis.

In the following chapters you will learn, in detail, how DB2 workload management can be used to address these business problems. You will also gain insight into the DB2 WLM concepts and architecture, how to get started using WLM with samples and best practices, tool integration as well as problem determination suggestions.



WLM architecture and features

In this chapter we describe the DB2 9.5 Workload Manager (WLM) architecture, its components, and how they interrelate with each other. The role of each major WLM component is discussed from a business point of view and what part it plays in the WLM architecture. Prior to DB2 V9.5, workload management meant the use of DB2 Query Patroller in conjunction with the DB2 Governor, while in DB2 9.5, workload management means the use of DB2 Workload Manager by itself, or in conjunction with DB2 Query Patroller and DB2 Governor.

We discuss the following topics in this chapter:

- ▶ DB2 Workload Manager concepts
- ▶ Architecture
- ▶ DB2 WLM monitor and control capabilities
- ▶ New database configuration parameter and catalog tables
- ▶ Working with WLM SQL and objects

2.1 DB2 Workload Manager concepts

In today's competitive business environment, the quest to increase business productivity creates an increasing need to do more with less resources, in an accelerated time frame and with less cost. A typical scenario for a data warehouse using DB2 would see multiple Extract, Transform, Load (ETL) jobs, multiple queries and reporting loads from multiple third-party Business Intelligence (BI) tools running throughout the day, and batch jobs and DBA utility jobs running all night. This would not take into account sudden shifts in priorities due to business needs, and perhaps the need to run multiple reports at the same time, creating sustained workloads during peak periods throughout the day. Sometimes, a "rogue" or runaway query may be submitted during peak hours, causing all workload in the system to slow down.

Some companies would acquire more hardware resources to address the problem, while others would terminate the resource hogs outright. Some would choose to tune the system performance to recover production capacity and others would create a very rigid approach to submitting workloads in the production system.

The use of Query Patroller and DB2 Governor has helped considerably in the management of these DB2 workloads. To extend and expand work management capabilities beyond those offered by these tools, a completely new design and architecture was created in the form of DB2 Workload Manager.

DB2 WLM is a powerful solution to address these multiple issues because there is a recognition that not all workloads can be tuned for optimum CPU and I/O usage in a very short span of time, and business users need a quick, flexible, and robust methodology to identify, manage, and control their workloads so as to maximize database server throughput and resource utilization.

The DB2 WLM architecture is primarily composed of the following components:

- ▶ Service classes
- ▶ Workloads
- ▶ Thresholds
- ▶ Work action sets
- ▶ Work classes

The DB2 WLM architecture revolves around addressing the problem of database resource assignment, namely CPU and I/O resources, to a DB2 workload. How can resource sharing be done effectively, and how is this resource sharing used to ensure stability to cope with changes in priority and fluctuating loads on the system?

2.1.1 DB2 service classes

A *DB2 service class* determines how you want to organize and group the work coming into the database. The DB2 service class acts as a unique execution environment for any grouping of work that you can assign resources to, control, and monitor. You can assign CPU or prefetch I/O priority resource to each of the DB2 service classes. All work in a database executes within a DB2 service class.

You can use the service class to organize activities in the database in a way that makes sense according to your business requirements. For example, if the database is used and shared by different business functions such as Sales, Marketing, and Customer Service, then you can create service superclasses for each of these functions. If within each function there are several departments that submit reports and load data, then service subclasses can be created for each of these departments. You can then monitor and control how each business unit can use the database resources.

Another example of categorizing work is to determine if the work coming in is Online Transaction Processing (OLTP), Online Analytical Processing (OLAP) or batch. Since the operating characteristics of an OLTP system are very different from that of an OLAP or batch system, the way in which these categories are controlled and monitored would also be different. In this case, you can designate one service class for OLTP, another service class for OLAP, and a third service class for batch. Different user groups can access a DB2 data warehouse using their own Business Intelligence (BI) reporting tools and Extract, Transform and Load (ETL) tools. In such a case, one way of setting up service classes is to designate one service class for each BI reporting tool and each ETL tool.

A DB2 service class can either be a *superclass* or a *subclass* within a superclass. This two-tier DB2 service class hierarchy provides a conceptual framework that closely resembles real-life situations and allows for orderly division of work among the DB2 service classes.

In DB2 9.5, when a database is created, DB2 creates three predefined default service superclasses:

- ▶ SYSDEFAULTUSERCLASS
- ▶ SYSDEFAULTSYSTEMCLASS
- ▶ SYSDEFAULTMAINTENANCECLASS

Each of the above superclasses has a default subclass SYSDEFAULTSUBCLASS.

Figure 2-1 shows a DB2 system with three user defined superclasses Sales, Finance and Marketing. The Sales superclass contains two service subclasses,

DirectSales and ChannelSales. The Finance and Marketing service superclasses have one subclass each, Accounting and Promotions respectively.

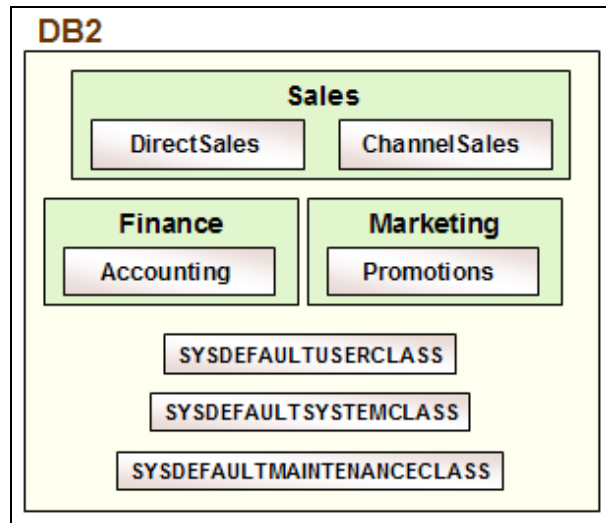


Figure 2-1 WLM service classes

Since the service class is the primary point for resource assignment to all incoming database requests and is used for monitoring database activities, we recommend that you identify service classes based on your critical business requirements. For example, you can set up service classes based on any of the following criteria:

- ▶ Service level agreements
- ▶ Need for very high-priority work to bypass normal work queue
- ▶ Conflict in sharing the same CPU and I/O resources
- ▶ Clearly defined logical work grouping
- ▶ Users or departments consistently exceeding resource constraints to the detriment of other users or departments
- ▶ Need to identify and analyze work contributing to resource peak usage
- ▶ Need to validate and plan for data server capacity

A DB2 service subclass can belong to only one DB2 service superclass but a DB2 service superclass can have one or more DB2 service subclasses. The maximum number of service superclasses that can be created for a database is 64. The maximum number of service subclasses that can be created under a superclass is 61.

You can create a DB2 service superclass using the CREATE SERVICE CLASS statement. Example 2-1 shows the SQL statements to create service superclasses and service subclasses as shown in Figure 2-1 on page 10.

Example 2-1 Create service superclasses

```

-----
-- Create service superclasses
-----
CREATE SERVICE CLASS sales;
CREATE SERVICE CLASS finance;
CREATE SERVICE CLASS marketing;
-----
-- Create service sub classes
-----
CREATE SERVICE CLASS directsales UNDER sales;
CREATE SERVICE CLASS channelsales UNDER sales;
CREATE SERVICE CLASS accounting UNDER finance;
CREATE SERVICE CLASS promotions UNDER marketing;
-----

```

In DB2 9.5 on AIX, DB2 workload management is integrated with the AIX Workload Manager such that the AIX Workload Manager can be used to control CPU shares for database work through the use of AIX service classes. This integration is discussed in more detail in Chapter 8, “AIX Workload Manager considerations” on page 193.

2.1.2 DB2 workloads

A *DB2 workload* is used to identify submitted database work or user connections so that it can be managed. A workload determines the source based on the database connection attributes under which the work is submitted. Each connection can be assigned to one and only one workload at any one time, but there can be multiple connections assigned to the same workload at the same time.

From a business perspective, identification is key because of the number of different ways that a user or system requests can come into the system. Many large IT installations now employ 3-tier or N-tier application servers, such as Websphere Application Server, where a user can access any of the application servers connected to the database at the same time. In other instances, there are data warehouse applications which allow access to the database server only through their own application server, and use only one generic user ID to access the database. DB2 WLM offers the means to be able to identify a user in a complex environment.

The ability to define multiple connection attributes for a single database connection allows for a robust environment where both simple and complex mapping to service classes can be easily handled.

The connection attributes tracked by a DB2 workload are:

- ▶ Application Name - Specified as APPLNAME in the workload definition statement.
- ▶ System authorization ID - Specified as SYSTEM_USER in the workload definition statement.
- ▶ Session authorization ID - Specified as SESSION_USER in the workload definition statement.
- ▶ Group of session authorization ID - Specified as SESSION_USER_GROUP in the workload definition statement.
- ▶ Role of session authorization ID - Specified as SESSION_USER_ROLE in the workload definition statement.
- ▶ Client user ID - Specified as CURRENT CLIENT_USERID in the workload definition statement.
- ▶ Client application name - Specified as CURRENT CLIENT_APPLNAME in the workload definition statement.
- ▶ Client workstation name - Specified as CURRENT CLIENT_WORKSTNNAME in the workload definition statement.
- ▶ Client accounting string - Specified as CURRENT CLIENT_ACCTNG in the workload definition statement.

These connection attributes determine how a user request is directed to a particular service class. Figure 2-2 shows two user defined workloads CAMPAIGN and NEWCAMPAIGN and the connection attributes used.

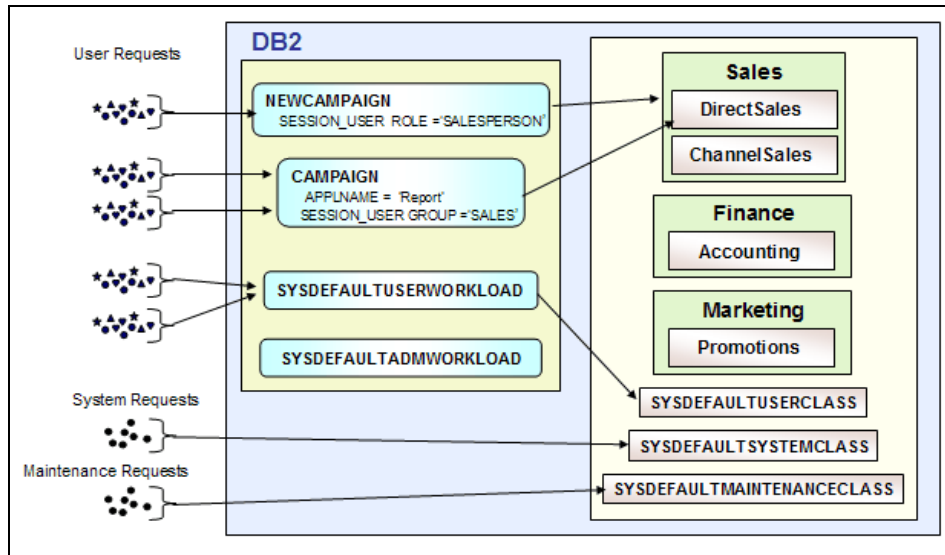


Figure 2-2 DB2 workload

Example 2-3 shows the SQL code to define the CAMPAIGN and NEWCAMPAIGN workloads.

Example 2-2 Create workloads

```
CREATE WORKLOAD campaign APPLNAME('Report') SESSION_USER GROUP ('SALES')
SERVICE CLASS directsales UNDER sales;
--
CREATE WORKLOAD newcampaign SESSION_USER ROLE ('SALESPERSON')
SERVICE CLASS sales POSITION BEFORE campaign;
```

A workload occurrence is started when the connection is attached to a workload definition. Any change in relevant connection attributes, workload definition, or USAGE privilege for a workload will cause the workload assignment to be reevaluated at the start of the next unit of work. If a new workload definition is assigned, the old workload occurrence is ended, and a new workload occurrence is started for the newly assigned workload definition.

If no match in connection properties is found, the user is assigned to the default workload, SYSDEFAULTUSERWORKLOAD. These users are directed to the SYSDEFAULTUSERCLASS service class. DB2 internal system requests such as DB2 prefetcher engine dispatchable units (EDUs), log reader EDUs, and log writer EDUs are directed to the SYSDEFAULTSYSTEMCLASS service class. DB2 internal maintenance requests such as DB2 asynchronous background

processing agents and DB2 Health monitor initiated utilities are directed to the SYSDEFAULTMAINTENANCECLASS service class.

The default system administration workload SYSDEFAULTADMWORKLOAD is a special DB2 workload that is not subject to any DB2 thresholds. This workload is primarily used by database administrators to perform their work or take corrective action.

For the workloads defined in the Example 2-2 on page 13, if user Bob runs the application with APPLNAME “Report” with a session authorization ID belonging to the group SALES, DB2 checks workload NEWCAMPAIGN first for a match before checking the workload CAMPAIGN, and then identifies Bob’s job as belonging to the workload CAMPAIGN. DB2 then directs Bob’s job to the DIRECTSALES service subclass.

2.1.3 DB2 thresholds

A DB2 WLM *threshold* is an object that sets a predefined limit over the consumption of a resource. In defining the threshold, a specified action can be triggered if the threshold is exceeded. The way a threshold works is similar to a trigger in that certain actions are initiated when a condition is reached. For example, thresholds can be used to limit the number of connections, the elapsed time, the amount of temp space used, and the estimated SQL cost of an activity.

There are two types of DB2 WLM thresholds:

- ▶ Activity thresholds: This threshold applies to an individual activity. When the resource usage of an individual activity violates the activity threshold, it triggers the threshold, which is applied only once.
- ▶ Aggregate threshold: This threshold sets a limit on a measurement across a set of multiple activities and operates as a running total, to which any work tracked by the threshold contributes.

The supported actions for a threshold are:

- ▶ STOP EXECUTION
Stop processing the activity that caused the threshold to be exceeded. For a threshold with a built-in queue, it means reject any newly arriving work from joining the queue.
- ▶ CONTINUE
Do not stop processing an activity if the threshold is exceeded. For a threshold with a built-in queue, it means add any newly arriving work to the queue.

► COLLECT ACTIVITY DATA

You can collect information about the activity that exceeded the threshold with different degrees of detail.

Figure 2-3 illustrates a threshold created to control database activities. In this example, when a new product is launched, the enthusiasm of entire Sales department spills over, and everyone on the Sales department wants to access the database. The Sales department wants to restrict the number of connections to 20 so that database performance is not affected by the surge in interest. This can be done by defining a threshold for service class SALES, setting the maximum number of concurrent database connections on a coordinator partition to 20, and the maximum number of queued connections to five. If these limits are exceeded, the application will be stopped.

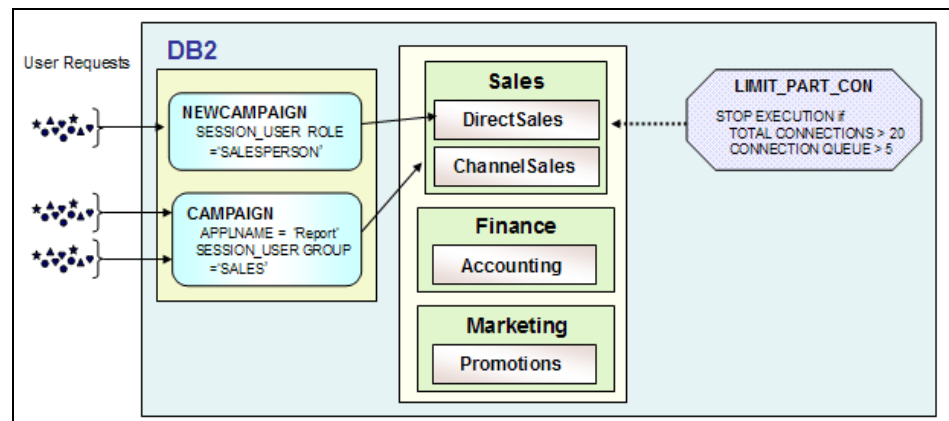


Figure 2-3 WLM threshold

Example 2-2 shows the threshold definition.

Example 2-3 Creating a threshold to limit partition connections

```
CREATE THRESHOLD limit_part_con
FOR SERVICE CLASS sales
ACTIVITIES ENFORCEMENT DATABASE PARTITION
WHEN TOTALSCPARTITIONCONNECTIONS > 20
AND QUEUEDCONNECTIONS > 5
COLLECT ACTIVITY DATA ON ALL WITH DETAILS
STOP EXECUTION;
```

Thresholds can be enforced on activities that are part of a threshold domain, which can range from the entire database to a single workload definition. For example, if the threshold domain is a database, an enforcement scope of the threshold may be just one database partition or all of the database partitions in

the database. Each threshold applies to a domain, which can range from the entire database to a single workload definition. The domain of a threshold defines the database object that the threshold is both attached to and operates on. Only engine work taking place within the domain of a threshold may be affected by it. The threshold domains are:

- ▶ Database
- ▶ Service superclass
- ▶ Service subclass
- ▶ Work action
- ▶ Workload

In each of these threshold domains, the threshold can be enforced over a single workload occurrence, a database partition, or across all the partitions of the database. This is known as the enforcement scope of the threshold. The enforcement scope can therefore be a workload occurrence, a database partition, or the entire database (also known as a global enforcement scope).

Figure 2-4 shows a summary of the DB2 WLM thresholds and the scope and domain that each threshold applies to.

Scope → Domain ↓	Database	Database Partition	Workload Occurrence
Database	<ul style="list-style-type: none"> ▪ Concurrent DB Coordinator Activities ▪ Estimated SQL Cost ▪ SQL Rows Returned ▪ Activity Total Time ▪ Connection Idle Time 	<ul style="list-style-type: none"> ▪ Total DB Partition Connections ▪ SQL System Temp Space 	N/A
Work Action Set	<ul style="list-style-type: none"> ▪ Concurrent DB Coordinator Activities ▪ Estimated SQL Cost ▪ SQL Rows Returned ▪ Activity Total Time 	<ul style="list-style-type: none"> ▪ SQL System Temp Space 	N/A
Service Super class	<ul style="list-style-type: none"> ▪ Concurrent DB Coordinator Activities ▪ Estimated SQL Cost ▪ SQL Rows Returned ▪ Activity Total Time ▪ Connection Idle Time 	<ul style="list-style-type: none"> ▪ Total Service Class Partition Connections ▪ SQL System Temp Space 	N/A
Service Sub class	<ul style="list-style-type: none"> ▪ Concurrent DB Coordinator Activities ▪ Estimated SQL Cost ▪ SQL Rows Returned ▪ Activity Total Time 	<ul style="list-style-type: none"> ▪ SQL System Temp Space 	N/A
Workload Definition	N/A	<ul style="list-style-type: none"> ▪ Number of Concurrent Workload Occurrences 	<ul style="list-style-type: none"> ▪ Concurrent Activities in a Workload Occurrence

Figure 2-4 DB2 WLM threshold summary

When multiple activity thresholds apply to one activity, you must decide which threshold to enforce and in what order. To resolve the scope of activity threshold resolution, WLM observes a hierarchy of domains so that a value defined in a

local domain overrides a value from a wider or more global domain. We follow the hierarchy of domains for these activity thresholds, numbered in the order in which threshold enforcement occurs:

1. Workload
2. Service subclass
3. Service superclass
4. Work Action
5. Database

Some thresholds, known as queueing thresholds, have a built-in queue and are defined with two boundaries: a threshold boundary and a queueing boundary. When a threshold boundary is reached, additional requests are added to the queue until the queueing boundary is reached. A queueing boundary defines an upper limit for the queue, beyond which a specified action, that is, STOP EXECUTION, is applied to any newly arriving work.

A threshold can be predictive or reactive. A predictive threshold means the threshold boundaries are checked before the tracked work is started. On the other hand, a reactive threshold means the boundaries are checked while the tracked work is executing.

2.1.4 Work class sets and work action sets

DB2 WLM provides you the capability to treat activities differently based on their activity type or some individual characteristic. For example, you want to treat stored procedures differently from all other read and write activity against the database. In one instance, you want to put a restriction on the number of concurrent Loads executing at any one time. You may also want DDL (Data Definition Language) put into a service class by itself. All the tasks listed above can be accomplished by using a *work action set*.

Work action sets allow us to apply DB2 thresholds with discrimination, and it can be used at the level of a DB2 service superclass to map to subclasses with discrimination.

Work action sets work hand in hand with *work class sets*. A work class set defines the characteristics of the work of interest, while the work action set dictates what is to happen when the work of interest is detected.

A work class has an associated work type. The supported work types are

- ▶ READ- for read related activities such as SELECT.
- ▶ WRITE - for update related activities such as DELETE, INSERT, UPDATE.
- ▶ CALL - for CALL statement.
- ▶ DML - for data manipulating activities such as SELECT, UPDATE, MERGE.

- ▶ DDL - for data definition activities such as CREATE, ALTER, COMMIT.
- ▶ LOAD - for LOAD utility.
- ▶ ALL - for all database activities.

Work actions are grouped into work action sets. A single work action can apply to either activities in the database or to activities in a service superclass, but not both.

To create a work action set and a work action, use the CREATE WORK ACTION SET statement. You must do the following:

- ▶ Associate the work action set with an existing work class.
- ▶ Associate the work action with either the database or an existing user-defined service superclass.

A work action set may be applied to an incoming database request, but more than one work action may be applied by the work action set. In such a case, the first matching work action in an ordered list will be applied. The position of the work action in the ordered list can be changed depending on which work action needs to be given priority.

The following actions can be performed within a DB2 work action set:

- ▶ Count activity
- ▶ Prevent execution of an activity
- ▶ Collect activity data
- ▶ Apply a DB2 threshold to a database
- ▶ Map work to a different service sub class within the same superclass
- ▶ Collect aggregate activity data for a service superclass

Figure 2-5 illustrates using work class set and work action set to manage workloads. Work class set CLASSIFY_WORK is associated with Sales work class and will classify the queries being identified by workload NEWCAMPAIGN and CAMPAIGN into four categories: expensive queries, moderately expensive, inexpensive queries, and expensive stored procedures. The work action set ACT_PLAN with three work actions is defined to dictate what happens when the work of interest is encountered. For expensive queries, stop the execution; for moderately expensive queries, collect data; for expensive stored procedures, send them to the Information Technology (IT) department for review.

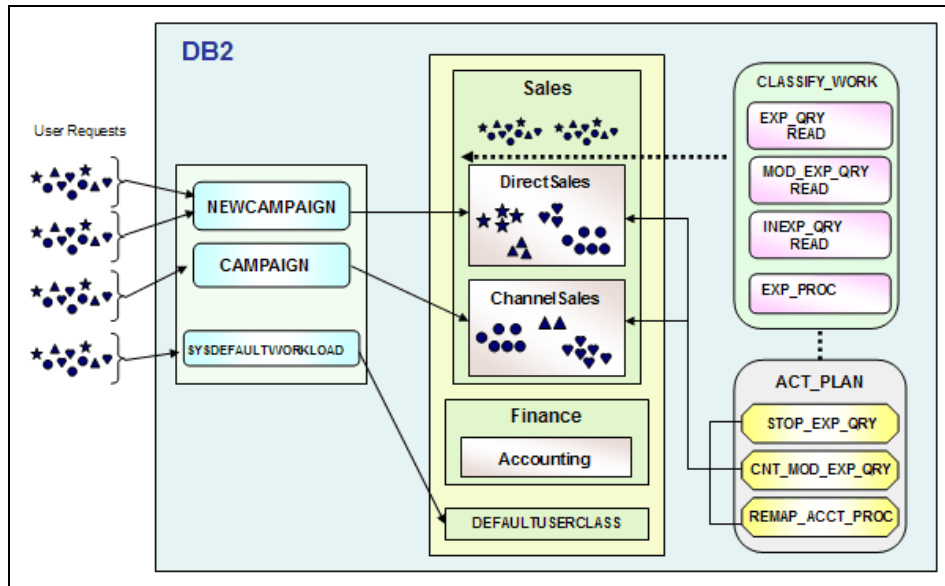


Figure 2-5 Work class set and work action set

Example 2-4 shows the definitions of work class set and work action set.

Example 2-4 Defining a work class set

```

-----
-- Create work class set
-----
CREATE WORK CLASS SET classify_work
(WORK CLASS exp_qry
  WORK TYPE READ FOR TIMERONCOST FROM 1000001 To UNBOUNDED,
 WORK CLASS mod_exp_qry
  WORK TYPE READ FOR TIMERON COST FROM 10000 TO 1000000,
 WORK CLASS inexp_qry
  WORK TYPE READ FOR TIMERON COST FROM 0 TO 10000
 WORK CLASS exp_proc
  WORK TYPE CALL ROUTINES IN SCHEMA "ACCOUNTING");
-----
-- Create work action set
-----
CREATE WORK ACTION SET act_plan FOR SERVICE CLASS sales
USING WORK CLASS SET classify_work
(WORK ACTION stop_exp_qry ON WORK CLASS exp_qry
  PREVENT EXECUTION COLLECT ACTIVITY DATA WITH DETAILS AND VALUES,
 WORK ACTION cnt_mod_exp_qry COUNT ACTIVITY,
 WORK ACTION remap_acct_proc ON WORK CLASS exp_proc
  MAP ACTIVITY WITH NESTED TO it_review);

```

2.2 Architecture

The architecture of the DB2 Workload Manager integrates all the workload management objects into a coherent whole in order to allow the ease of identifying, managing, and monitoring all the workload in the DB2 database. The workload on the system can be analyzed to determine how the system can be designed to cope with the current and anticipated workload. Performance monitoring using DB2 WLM can track the behavior of the system either on a granular level or over a wide-ranging period. The principal benefit, however, is the ability to understand the characteristics of the incoming workload, and that knowledge will enable you to manage and maintain the system desired response times and throughput. In addition, some of the most vexing problems in a database environment, such as runaway or rogue queries and agent contention can be handled better with the new WLM capabilities.

The process of using DB2 WLM effectively starts with analyzing the workload by identifying the types and frequency of workloads and then creating service classes in order to classify the workload into manageable groups.

The diagram in Figure 2-6 illustrates the interrelationships of the main components of the DB2 WLM architecture.

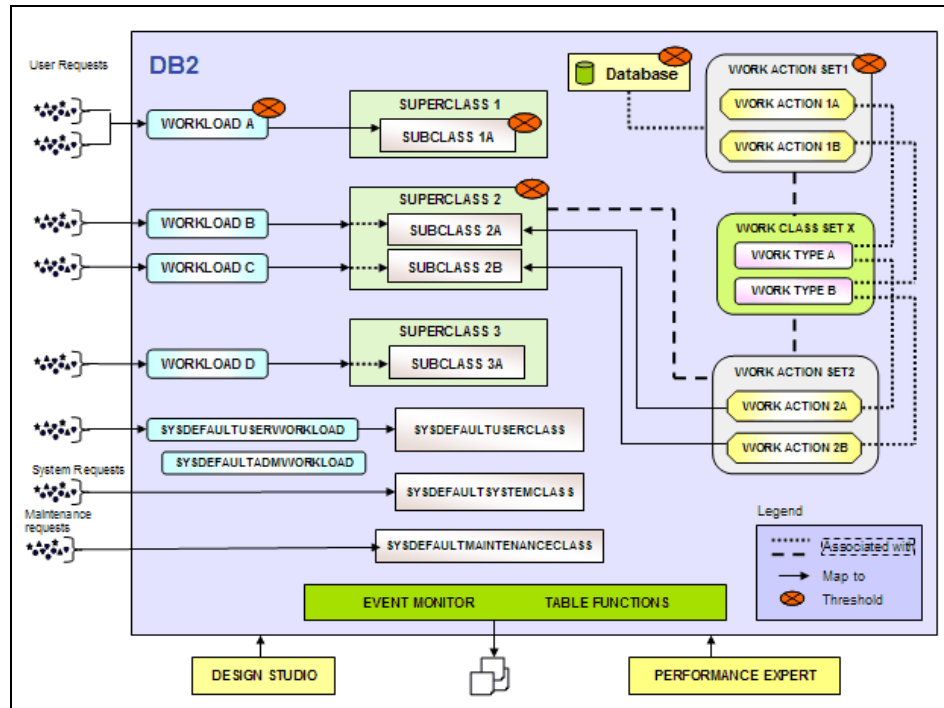


Figure 2-6 DB2 WLM architecture and AIX WLM

A user makes a database connection and is assigned to a workload. All activities running under the workload occurrence are mapped to service classes.

In Figure 2-6, users assigned to workload A are mapped to SUPERCLASS1 by specifying the UNDER keyword for the SERVICE CLASS keyword. The connection belongs to service superclass 1, but all activities issued out of the connection are automatically mapped to service subclass 1A.

Users assigned to workloads B and C are mapped to service superclass 2. Any work submitted for workload occurrences belonging to workloads B and C can be mapped to service subclasses 2A and 2B. All activities that are mapped to SUPERCLASS 2 and match a work class in work class set X to which a MAP ACTIVITY work action is associated are mapped to service subclass 2A or 2B as specified by the work action.

A work action set can be defined for either a database or a service superclass. In the diagram, work actions 1A and 1B belong to work action set 1, and it is defined for a database. Work actions 1A and 1B can be any one of the following actions:


- ▶ A threshold
- ▶ PREVENT EXECUTION
- ▶ COLLECT ACTIVITY DATA
- ▶ COUNT ACTIVITY

Work actions 2A and 2B belong to work action set 2, and they are defined for service superclass 2. The work actions 2A and 2B can be any of the following actions:

- ▶ A mapping action mapping an activity to any service subclass in service superclass 2 except for the default service subclass.
- ▶ PREVENT EXECUTION
- ▶ COLLECT ACTIVITY DATA
- ▶ COLLECT AGGREGATE ACTIVITY DATA
- ▶ COUNT ACTIVITY

Users assigned to workload D are mapped to a service superclass 3, which does not have a user-defined service subclass. In this case, the connections are mapped to the default subclass SYSDEFAULTSUBCLASS of service superclass 3.

Connections that do not map to a user-defined workload are mapped to the default workload SYSDEFAULTUSERWORKLOAD, and this in turn is mapped to the default service superclass for user requests, SYSDEFAULTUSERCLASS. Internal DB2 system connections are mapped to the default service superclass for internal DB2 connections, SYSDEFAULTSYSTEMCLASS, while internal DB2 maintenance connections are mapped to the default service superclass for maintenance requests, SYSDEFAULTMAINTENANCECLASS.

As illustrated in Figure 2-6, DB2 WLM thresholds (indicated by ) can be defined on any or all of the following:

- ▶ Database
- ▶ Work action set
- ▶ Service superclass
- ▶ Service subclass
- ▶ Workload

If the DB2 environment is on AIX, and the AIX WLM is being used, it is possible to associate the DB2 service classes with their corresponding AIX service classes as illustrated in Figure 2-7. The DB2 service classes are associated with their corresponding AIX service classes by using the OUTBOUND CORRELATOR keyword in the CREATE SERVICE CLASS statement to associate threads from the DB2 service class to an AIX service class.

In Figure 2-7, the DB2 service superclasses 1 and 2 are associated with AIX WLM service classes `_DB2_SUPERCLASS 1` and `_DB2_SUPERCLASS 2`, respectively. The DB2 service subclasses 1A, 2A and 2B are associated with AIX WLM subclasses `_DB2_SUBCLASS 1A`, `_DB2_SUBCLASS 2A` and `_DB2_SUBCLASS 2B`, respectively. The DB2 service class `SYSDEFAULTUSERCLASS` is associated with AIX WLM service class `_DB2_DEF_USER`, and the DB2 SERVICE classes `SYSDEFAULTSYSTEMCLASS` and `SYSDEFAULTMAINTENANCECLASS` are associated with AIX WLM service class `_DB2_DEF_SYS`.

We discuss the relationship between AIX WLM and DB2 WLM in more detail in Chapter 8, “AIX Workload Manager considerations” on page 193.

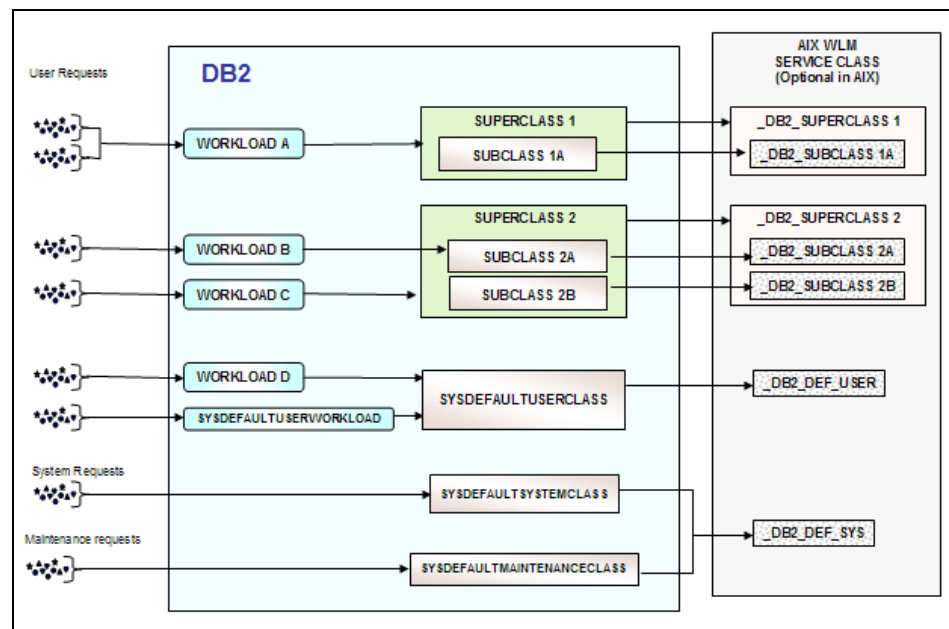


Figure 2-7 DB2 WLM integrated with AIX WLM

The following lists shows all the WLM-exclusive SQL that can be used to set up and manage WLM:

- ▶ CREATE WORKLOAD, ALTER WORKLOAD, DROP WORKLOAD
- ▶ GRANT (Workload Privileges), REVOKE (Workload Privileges)
- ▶ CREATE SERVICE CLASS, ALTER SERVICE CLASS, DROP SERVICE CLASS
- ▶ CREATE WORK CLASS SET, ALTER WORK CLASS SET, DROP WORK CLASS SET

- ▶ CREATE WORK ACTION SET, ALTER WORK ACTION SET, DROP WORK ACTION SET
- ▶ CREATE THRESHOLD, ALTER THRESHOLD, DROP THRESHOLD
- ▶ CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE, DROP HISTOGRAM TEMPLATE

We discuss the SQL statements in more detail in 2.5, “Working with WLM SQL and objects” on page 30.

In creating the WLM objects and the SQL to generate it, use the DWE Design to help generate SQL and rules validation for WLM. A section on how to use this tool is discussed in Chapter 10, “DB2 WLM and DWE Design Studio” on page 235.

2.3 DB2 WLM monitor and control capabilities

This section describes the DB2 WLM monitoring and control capabilities for real-time and historical aggregate data. DB2 9.5 provides new table functions for direct ad-hoc querying of WLM objects or obtaining summarized statistics over time. The event monitor has been enhanced to support workload management. In addition, DB2 9.5 offers new stored procedures to cancel activities, capture information about individual activities, and collect and reset statistics for workload management objects.

2.3.1 Real-time monitoring

In DB2 WLM, real-time monitoring and statistical monitoring capabilities are built into DB2 using the SYSPROC schema and can be accessed with little impact on currently executing workloads. Real-time monitoring is accomplished by using DB2 table functions to obtain operational information. The following DB2 9.5 new table functions are for real-time monitoring:

- ▶ **WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES**
This table function returns a list of workload occurrences across database partitions assigned to a service class, information about the current state, the connection attributes used to assign the workload to the service class, and activity statistics indicating activity volume and success rates.
- ▶ **WLM_GET_SERVICE_CLASS_AGENTS**
This table function returns a list of database agents associated with a service class or application handle, the current state of the agent, the action the agent is performing and the status of that action.

- ▶ **WML_GET_WORKLOAD_OCCURRENCE_ACTIVITIES**
This table function returns a list of current activities associated with a workload occurrence, information about the activity, the type of activity and the time the activity started.
- ▶ **WLM_GET_ACTIVITY_DETAILS**
This table function returns detail about an individual activity, the activity type, and additional information pertinent to that activity type.

2.3.2 Statistics table functions

The WLM table functions can also be used to obtain statistics about DB2 workload manager objects. Statistics are maintained for service classes, work classes, workloads, and threshold queues. These statistics are resident in memory and can either be viewed in real-time using WLM statistics table functions, or the statistics can be collected and sent to a statistics event monitor where they can be viewed for historical analysis.

The following are the WLM statistics table functions:

- ▶ **WLM_GET_SERVICE_SUPERCLASS_STATS**
This table function returns information about the concurrent connection high watermark that was calculated since the last statistics reset.
- ▶ **WLM_GET_SERVICE_SUBCLASS_STATS**
This table function returns summarized statistics such as number of activities and average execution time calculated since the last time reset.
- ▶ **WLM_GET_WORKLOAD_STATS**
This table function returns summarized statistics for one or all workloads and database partitions.
- ▶ **WLM_GET_WORK_ACTION_SET_STATS**
This table function returns summarized statistics for one or work action sets across one or more database partitions.
- ▶ **WLM_GET_QUEUE_STATS**
This table function returns information about threshold queues.

The workload management table functions can also be used in conjunction with the snapshot monitor table functions to aid in problem solving or performance tuning.

2.3.3 Event monitors for DB2 WLM

The enhanced DB2 event monitor provides the capability to monitor WLM-specific events. A total of seventy one monitor elements are available to provide information about a workload management implementation. WLM event monitors capture a set of events for debugging or collect historical information for subsequent analysis. On the other hand, table functions look at and gather point-in-time information.

Unlike the non-WLM event monitors, the WLM event monitors don't have event conditions which can be triggered by the WHERE clause of the CREATE EVENT MONITOR statement. The WLM event monitors are dependent on how the attributes of service classes, workloads, work classes, and thresholds are set to send activity or aggregate information to the WLM monitors.

There are three types of WLM event monitors:

- ▶ **Activities**
This type of event monitor is used to collect information about an activity.
- ▶ **Threshold violations**
This type of event monitor captures information whenever a threshold violation occurs.
- ▶ **Statistics**
This type of event monitor captures statistics over a set time frame. This event monitor gathers aggregated activity information and is directed to a single service class or work class.

A sample script *wlmevmon.ddl* in the `~/sqllib/misc` directory shows how to create and enable three event monitors DB2ACTIVITIES, DB2STATISTICS and DB2THRESHOLDVIOLATIONS.

You can collect various detail levels of the full set of statistics, including average execution times, average queueing times, and histograms. The statistics level to be gathered are specified as an option in the service subclass or work class:

- ▶ COLLECT AGGREGATE ACTIVITY DATA BASE
- ▶ COLLECT AGGREGATE ACTIVITY DATA EXTENDED
- ▶ COLLECT AGGREGATE REQUEST DATA BASE

There are statistics that is maintained on the given WLM objects on each database partition regardless of whether COLLECT AGGREGATE ACTIVITY DATA or COLLECT AGGREGATE REQUEST DATA was specified or not. These statistics are listed as follows:

- ▶ **Threshold queues:**
 - Total queue assignments

- Top queue size top.
- Total queue time.
- ▶ Service subclasses:
 - High watermark concurrent activity
 - Total coordinator activities completed
 - Total coordinator activities aborted
 - Total coordinator activities rejected total
 - Number of active requests
- ▶ Service superclasses:
 - Concurrent connection top
- ▶ Workloads:
 - Total workload occurrences completed
 - High water mark of concurrent workload occurrences
 - High water mark of Concurrent activity
 - Total coordinator activities completed total
 - Total coordinator activities aborted total
 - Total Coordinator activities rejected total
 - Total workload occurrences completed
- ▶ Work class through a work action
 - Total activities

The following statistics are collected when a service subclass or a work class is created or altered with the option COLLECT AGGREGATE ACTIVITY DATA BASE:

- ▶ Coordinator activity lifetime average
- ▶ Average coordinator activities execution time average
- ▶ Coordinator activity queue time average
- ▶ High watermark of cost estimate
- ▶ High watermark of Actual rows returned
- ▶ High watermark of temporary table space
- ▶ Activity lifetime histogram
- ▶ Activity execution time histogram
- ▶ Activity queue time histogram

The following statistics are collected for each database partition for the corresponding service class or work class when a service subclass or a work class is created or altered with the option COLLECT AGGREGATE ACTIVITY DATA EXTENDED:

- ▶ Non-nested coordinator activity inter-arrival time
- ▶ Coordinator activity estimated cost average
- ▶ Activity inter-arrival histogram

► Activity estimated cost histogram

The following statistics are collected for each database partition for the corresponding service subclass when a service subclass or a work class is created or altered with the option COLLECT AGGREGATE REQUEST DATA BASE:

- Request execution time average
- Request execution time histogram

Histogram

A *histogram* is defined as a graphical display of tabulated frequencies. In DB2 WLM, the histogram is represented by a collection of bins or rectangles where the width is represented by a range of values, and the height is represented by the count or frequency of these values. DB2 WLM histograms have a fixed number of 41 bins. Figure 2-8 shows a histogram example.

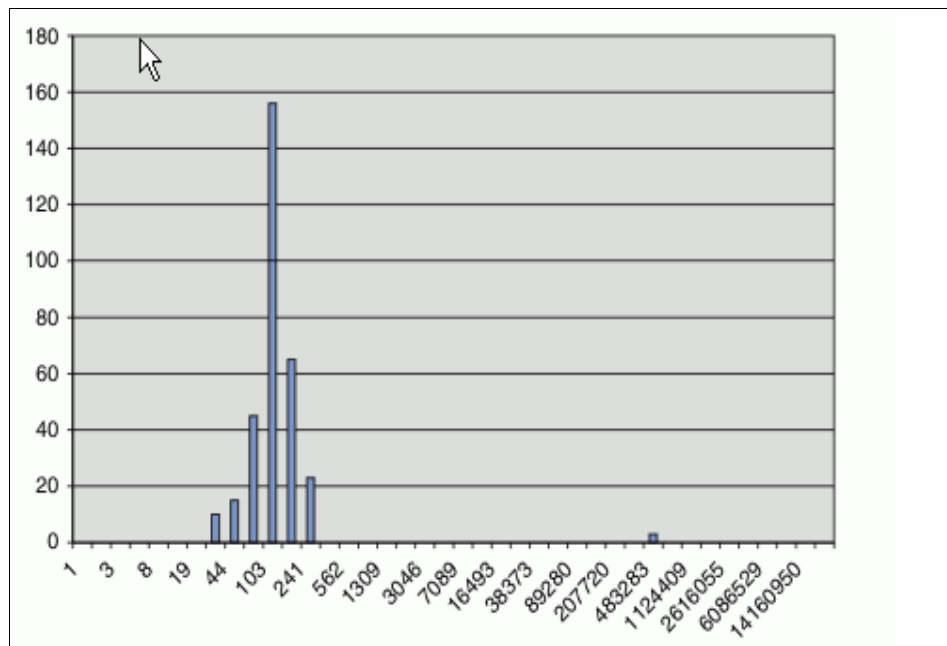


Figure 2-8 Histogram plotted to a bar chart

Histograms can be used to discover workload situations that would not be obvious when looking at the data alone. The distribution of values and the outlying values can be determined at a glance. When histograms are applied to a partitioned database environment, the histogram bins can be used to analyze the

distribution of values per partition, or they can be combined into one histogram to get a global view of the data.

Histograms are available for service subclasses and work classes and are collected when any of the following clauses are specified when creating or altering the object:

- ▶ COLLECT AGGREGATE ACTIVITY BASE
- ▶ COLLECT AGGREGATE ACTIVITY BASE EXTENDED

A histogram template can also be created to describe the high bin values for each of the histograms that are collected for an object. These histogram templates are objects with no predefined measurement units that specify what a histogram should look like.

2.3.4 WLM stored procedures

DB2 WLM stored procedures are provided to cancel an activity, capture information about an individual activity, and collect and reset statistics for workload management objects.

The following is a list of these stored procedures:

- ▶ WLM_CANCEL_ACTIVITY - cancels a given activity.
- ▶ WLM_CAPTURE_ACTIVITY_IN_PROGRESS - gathers information on a given activity including all its child activities and writes it to the active activities event monitor
- ▶ WLM_COLLECT_STATS - gathers and resets statistics on service classes, work classes and threshold queues and writes it to the statistics event monitor.
- ▶ WLM_SET_CLIENT_INFO - allows you to set the values of any of the client information fields at the DB2 server using a CALL statement.

2.4 New database configuration parameter and catalog tables

A new database configuration parameter and several catalog tables are introduced to support WLM.

WLM_COLLECT_INT

This new database configuration parameter WLM Collection Interval is used to specify a collection and reset interval, in minutes, for workload management

statistics. This parameter is only specified on the catalog partition and will determine how often workload management statistics are collected and sent to any statistics event monitor.

All WLM statistics table functions will return the accumulated statistics for the specified interval until the next reset. The WLM_COLLECT_STATS procedure will perform the same collect and reset operations that would occur automatically on the interval defined by the WLM_COLLECT_INT database configuration parameter.

New catalog tables

The new WLM-specific system catalog views are:

- ▶ SYSCAT.HISTOGRAMTEMPLATEBINS - Each row represents a histogram template bin.
- ▶ SYSCAT.HISTOGRAMTEMPLATES - Each row represents a histogram template.
- ▶ SYSCAT.HISTOGRAMTEMPLATEUSE - Each row represents a relationship between a workload management object that can use histogram templates and a histogram template.
- ▶ SYSCAT.SERVICECLASSES - Each row represents a service class.
- ▶ SYSCAT.THRESHOLDS - Each row represents a threshold.
- ▶ SYSCAT.WORKACTIONS - Each row represents a work action.
- ▶ SYSCAT.WORKACTIONSETS - Each row represents a work action set
- ▶ SYSCAT.WORKCLASSES - Each row represents a work class.
- ▶ SYSCAT.WORKCLASSSETS - Each row represents a work class set.
- ▶ SYSCAT.WORKLOADAUTH - Each row represents a user, group or role that has been granted USAGE privilege on a workload.
- ▶ SYSCAT.WORKLOADCONNATTR - Each row represents a connection attribute in the definition of a workload
- ▶ SYSCAT.WORKLOADS - Each row represents a workload.

2.5 Working with WLM SQL and objects

In this section, we introduce the new SQL statement for DB2 WLM objects service classes, workload, threshold, and work classes. For the details of the SQL statements, refer to DB2 documents:

- ▶ DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp>

Author Comment: Update link (wjc)

- ▶ DB2 manuals:
 - SQL Reference, Volume 1, SC10-4249*
 - SQL Reference, Volume 2, SC10-4250*

2.5.1 DB2 Service classes

Use the CREATE SERVICE CLASS statement to define the service superclass and service subclass.

When you use the CREATE SERVICE CLASS statement, you need to specify the name of the service class. Optionally, you can specify the following properties:

- ▶ AGENT PRIORITY clause
 - Use this clause to control agent priority (CPU).
- ▶ PREFETCH PRIORITY clause
 - Use this clause to control prefetcher priority (Prefetcher I/O).
- ▶ COLLECT AGGREGATE ACTIVITY DATA/COLLECT AGGREGATE REQUEST DATA clause
 - Use this clause to collect statistics information for service subclass only.
- ▶ COLLECT ACTIVITY DATA clause
 - Use this clause to collect activity information for service subclass only.
- ▶ OUTBOUND CORRELATOR clause
 - Use this clause to associate the DB2 service class to an AIX service class.
- ▶ ENABLE or DISABLE clause
 - Use this clause to specify whether or not connections and activities can be mapped to the service class.

When you create a service superclass, DB2 automatically creates a default service subclass SYSDEFAULTSUBCLASS under it. When you create the service subclass, you need to specify the name of the parent service superclass.

Example 2-5 shows the CREATE SERVICE CLASS statement to create a service superclass.

Example 2-5 Creating a service superclass

```
CREATE SERVICE CLASS sales AGENT PRIORITY -20
```

Example 2-6 shows the CREATE SERVICE CLASS statements creating two service subclasses.

Example 2-6 Creating a service subclass

```
CREATE SERVICE CLASS sales_read UNDER sales PREFETCH PRIORITY HIGH  
CREATE SERVICE CLASS sales_write UNDER sales PREFETCH PRIORITY LOW
```

DB2 automatically creates a default subclass under each service superclass. We recommend issuing a COMMIT after a WLM-exclusive statement to write the changes to the system catalog.

2.5.2 DB2 Workloads

Use the CREATE WORKLOAD statement to define the workload. When you use the CREATE WORKLOAD statement, you need to specify the name of the workload and the attribute of the connection.

The following are clauses to specify the attribute of the connection.

▶ **APPLNAME**

You need to know your application name as known to the database server. Your application name is shown in the “Application name” field in system monitor output and in the output from the LIST APPLICATIONS command.

▶ **SYSTEM_USER**

You can use the user that connects to the database.

▶ **SESSION_USER**

If you use the SET SESSION AUTHORIZATION statement to change the current session user, you can use this attribute. If you do not use the SET SESSION AUTHORIZATION statement, the current session user is the same as the system user.

▶ **SESSION_USER GROUP**

You can use the group name which the current session user belongs to.

▶ **SESSION_USER ROLE**

If you use the roles to simplify privilege management, you can use this attribute.

▶ **CURRENT_CLIENT_USERID**

You can use the client user ID from the client information specified for the connection.

▶ CURRENT CLIENT_APPLNAME

You can use the application name from the client information specified for the connection.

▶ CURRENT CLIENT_WRKSTNNAME

You can use the workstation name from the client information specified for the connection.

▶ CURRENT CLIENT_ACCTNG

You can use the accounting string from the client information specified for the connection. The accounting string basically is a custom tag that can be used to identify a set of activities based on user-defined attributes such as a custom report.

Users can connect directly or through an application server with different connection properties to DB2. DB2 WLM uses these properties to direct them to the appropriate service class. In an N-tier client-server environment, an application server can use the *sqlseti* API, where client information is set on the client side, to pass specific client information to the DB2 data server. Another way to set client information is to use the WLM_SET_CLIENT stored procedure to set client information for the connection at the DB2 server.

Optionally, you can specify the following attributes or properties:

▶ Workload attributes

– SERVICE CLASS

Use this clause to assign work to a service class.

– ALLOW DB ACCESS or DISALLOW DB ACCESS

Use this clause to specify whether a request is allowed access to the database or not.

– ENABLE or DISABLE

Use this clause to specify whether or not this workload will be considered when a workload is chosen.

▶ COLLECT ACTIVITY DATA clause

Use this clause to collect activity information for workload.

▶ POSITION clause

Use this clause to specify the workload order DB2 searched.

Example 2-7 shows the CREATE WORKLOAD statement to create a workload assigned to a service superclass.

Example 2-7 Creating a workload

```
CREATE WORKLOAD sales_wl applname('sales.exe') SERVICE CLASS sales
```

The SALES_WL workload is associated with requests executed from the application name sales.exe. These requests are run in the SALES service superclass. If you do not specify the SERVICE CLASS clause, these requests are run in the default service class SYSDEFAULTUSERCLASS.

Workloads has evaluation order specified by the POSITION keyword. If the POSITION keyword is not specified, the new workload is positioned after all the other defined workloads, but before the last workload, SYSDEFAULTUSERWORKLOAD.

2.5.3 DB2 Thresholds

Use the CREATE THRESHOLD statement to define the threshold.

When you use the CREATE THRESHOLD statement, you need to specify the following properties:

- ▶ Name of the threshold
- ▶ Threshold domain
 - DATABASE / SERVICE CLASS / WORKLOAD
- ▶ Enforcement scope
 - DATABASE / DATABASE PARTITION / WORKLOAD OCCURRENCE
- ▶ Threshold
 - Elapsed time (ACTIVITYTOTALTIME)
The maximum amount of time that the data server should spend processing an activity.
 - Idle time (CONNECTIONIDLETIME)
The maximum amount of time that a connection can be idle.
 - Estimated cost (ESTIMATEDSQLCOST)
The maximum estimated cost permitted for DML
 - Rows returned (SQLROWSRETURNED)
The maximum number of rows that can be returned for DML.
 - Temporary space (SQLTEMPSPACE)

The maximum amount of temporary table space that can be used by a DML activity at any database partition.

- Concurrent workload occurrences (CONCURRENTWORKLOADOCCURRENCES)

The maximum number of workload occurrences that can run concurrently on the coordinator partition.

- Concurrent workload activities (CONCURRENTWORKLOADACTIVITIES)

The maximum number of coordinator and nested activities that can run concurrently in a workload occurrence.

- Concurrent database activities (CONCURRENTDBCOORDACTIVITIES)

The maximum number of concurrent coordinator activities across all database partitions.

- Total database partition connections (TOTALDBPARTITIONCONNECTIONS)

The maximum number of concurrent database connections on a coordinator partition for a database.

- Total service class partition connections (TOTALSCPARTITIONCONNECTIONS)

The maximum number of concurrent database connections on a coordinator partition for a service superclass.

- ▶ Action
 - STOP EXECUTION / CONTINUE

Optionally, you can specify the following properties:

- ▶ Action
 - COLLECT ACTIVITY DATA clause
- ▶ ENABLE or DISABLE

Use this clause to specify whether or not the threshold is enabled for use by the database manager.

Example 2-8 shows the CREATE THRESHOLD statement to create a threshold assigned to a service subclass.

Example 2-8 Creating a threshold

```
CREATE THRESHOLD limit_cost for SERVICE CLASS sales ACTIVITIES ENFORCEMENT
database WHEN estimatedsqlcost > 10000 STOP EXECUTION
```

The threshold (when `estimatedsqlcost > 10000`) is enforced for activity which runs in the department superclass across all database partitions.

2.5.4 DB2 work classes

Use the `CREATE WORK CLASS SET` to create a work class set.

There are two ways of creating a work class:

- ▶ Use the `CREATE WORK CLASS SET` statement to create a new work class set to contain the new work class.
- ▶ Add a new work class to an existing work class set using the `ALTER WORK CLASS SET` statement.

When you use the `CREATE WORK CLASS SET` statement, you need to specify the following properties:

- ▶ Name of the work class set
- ▶ Work class definition

Table 2-1 shows the type keywords available for work classes and the SQL statement corresponding to the different keywords. Except for the `LOAD` command, all the statements in Table 2-1 are intercepted immediately before execution.

Table 2-1 Work type keywords and associated SQL statements

Work type keyword	Applicable SQL statements
READ	<ul style="list-style-type: none"> ▶ All <code>SELECT</code> statements (select into, values into, full select) ▶ <code>SELECT</code> statements containing a <code>DELETE</code>, <code>INSERT</code>, or <code>UPDATE</code> are not included ▶ All <code>XQuery</code> statements
WRITE	<ul style="list-style-type: none"> ▶ All <code>UPDATE</code> statements (searched, positioned) ▶ All <code>DELETE</code> statements (searched, positioned) ▶ All <code>INSERT</code> statements (values, subselect) ▶ All <code>MERGE</code> statements ▶ All <code>SELECT</code> statements containing a <code>DELETE</code>, <code>INSERT</code>, or <code>UPDATE</code> statement
CALL	<ul style="list-style-type: none"> ▶ <code>CALL</code> statement ▶ The <code>CALL</code> statement is only classified under the <code>CALL</code> and <code>ALL</code> work class types.
DML	All statements that are classified under the <code>READ</code> and <code>WRITE</code> work class types

Work type keyword	Applicable SQL statements
DDL	<ul style="list-style-type: none"> ▶ All ALTER statements ▶ All CREATE statements ▶ COMMENT statement ▶ DECLARE GLOBAL TEMPORARY TABLE statement ▶ DROP statement ▶ FLUSH PACKAGE CACHE statement ▶ All GRANT statements ▶ REFRESH TABLE ▶ All RENAME statements ▶ All REVOKE statements ▶ SET INTEGRITY statement
LOAD	<ul style="list-style-type: none"> ▶ Load utility. ▶ The load utility is only classified under the LOAD and ALL work class types.
ALL	<ul style="list-style-type: none"> ▶ All database activity. ▶ If the action is a threshold, the database activity that the threshold is applied to depends on the type of threshold. For example, if the threshold type is ESTIMATEDSQLCOST, only DML activity with an estimated cost (in timerons) is affected by the threshold.

Example 2-9 shows the CREATE WORK CLASS SET statement to create a work class READ_WORK and a work class WRITE_WORK.

Example 2-9 Creating a work class

```
CREATE WORK CLASS SET sales_work
(WORK CLASS read_work WORK TYPE read,
WORK CLASS write_work WORK TYPE write)
```

2.5.5 DB2 work action set

Use the CREATE WORK ACTION SET to define a work action set. You must associate the work action set with an existing work class set. In addition, you must also associate the work action set with the database or an existing service superclass.

There are two ways of creating a work action:

- ▶ Use the CREATE WORK ACTION SET statement to create a new work action.
- ▶ Add a new work action to an existing work action set using the ALTER WORK ACTION SET statement.

When you use the CREATE WORK ACTION SET statement, you need to specify the following properties:

- ▶ Name of the work action set
- ▶ FOR DATABASE / SERVICE CLASS

This clause specifies the database manager object to which the actions in this work action set will apply.
- ▶ USING WORK CLASS SET work-class-set-name

This clause specifies the work class set containing the work classes that will classify database activities on which to perform actions.
- ▶ WORK ACTION work-action-name on WORK CLASS work-class-name

This clause specifies the work action definitions including the following:

 - MAP ACTIVITY WITH NESTED/WITHOUT NESTED TO service-subclass-name
 - WHEN
 - CONCURRENTDBCOORDACTIVITIES / AND QUEUEDACTIVITIES
 - SQLTEMPSPACE
 - SQLROWSRETURNED
 - ESTIMATEDSQLCOST
 - ACTIVITYTOTALTIME
 - COLLECT ACTIVITY DATA NONE/ COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
 - STOP EXECUTION / CONTINUE
- ▶ ENABLE or DISABLE

This clause activates or deactivates the work action.
- ▶ ACTIVITY LIFETIME / QUEUE TIME / EXECUTETIME / ESTIMATEDCOST / INTERARRIVAL HISTOGRAM TEMPLATE template-name

These are properties of histogram templates to be used when collecting aggregate activity data for activities associated with the work class to which this work action is assigned.

Example 2-10 shows the CREATE WORK ACTION SET statement to create a work action set SALES_ACTION that is associated with the service superclass SALES.

Example 2-10 Creating a work action set.

```
CREATE WORK ACTION SET sales_action FOR SERVICE CLASS sales
  USING WORK CLASS SET sales_work
```

```
(WORK ACTION read_action ON WORK CLASS read_work MAP ACTIVITY TO
sales_read,
  WORK ACTION write_action ON WORK CLASS write_work MAP ACTIVITY TO
sales_write)
```



Getting started

DB2 Workload Management (WLM) provides you the capability to simplify the overall complexity of database workload management and monitoring. Implementing and setting up WLM is fairly easy once you are familiar with the concepts of WLM. To fully utilize the capability of WLM, some special considerations are recommended in designing and setting up your database environment. If you are new to DB2 WLM, you can start from the default settings DB2 provides and gradually explore the full function of DB2 WLM to take full control over your database environment and applications.

In this chapter we discuss the following topics:

- ▶ System requirements
- ▶ Planning DB2 environment
- ▶ Lab environment
- ▶ Installing DB2
- ▶ First steps

3.1 System requirements

In this section we discuss the hardware and software requirements for DB2 9.5 and WLM. In general, the system requirements to use DB2 WLM are the same as the system requirements for DB2 9.5 with a few exceptions. When there are special considerations for WLM, we note them along our discussion.

Depending on your environment, you might want to reevaluate your CPU and memory requirements once you have the WLM in place. With WLM, you'll be able to manage your DB2 workloads in more granular level and thus can fully utilize your hardware. You might discover that you can use less powerful hardware than you originally planned.

3.1.1 Hardware

There are no special hardware requirements for WLM. Plan your hardware based on your database requirements. The minimum hardware requirements common for all operating systems running DB2 9.5 for Linux, UNIX, and Windows are:

- ▶ DB2 without any graphical tools needs minimum 256 MB memory. For graphical tools and improved performance, 1.0 GB RAM is recommended. These requirements do not include any additional applications nor operating system memory requirements.
- ▶ For CPU speed, minimum CPU 1.0 GHz is required. For better performance 2.0 GHz is recommended.
- ▶ Disk space depends on the type of installation you are going to choose. Default installation requires approximately 1.0 GB for DB2 software. Add additional 256 MB disk space per instance. Sufficient disk space for your databases is needed as well.
- ▶ If you are going to collect data through WLM event monitors for historical analysis, you might want to allocate additional disk space for a separated file system for the event monitor data.

3.1.2 Software

DB2 WLM for Linux, UNIX, and Windows is available on all of the platforms that support DB2 9.5.

If you want to utilize operating system workload management capability, AIX WLM is supported on DB2 9.5.

3.1.3 Platforms supported

The section lists the supported version of Linux, UNIX, and Windows operating systems. We recommend you always check the latest required patch levels and software upgrades from:

<http://www-306.ibm.com/software/data/db2/9r2/sysreqs.html>

Author Comment: The URL needs to be fixed to whatever it will be

Linux and UNIX

The following Linux and UNIX platforms support WLM:

- ▶ AIX
 - AIX 5.3 (eServer™ pSeries®, IBM System p™, and IBM System p5™)
- ▶ HP-UX
 - HP-UX 11iv2 on Itanium® based HP Integrity Series Systems
 - HP-UX 11iv3 on Itanium based HP Integrity Series Systems
- ▶ Linux x86
 - Red Hat Enterprise Linux 4 or 5
 - SUSE Linux Enterprise Server 9 or 10 on x86 Intel®(R) Pentium®(R), Intel Xeon®(R), and AMD™ Athlon™
- ▶ Linux x64
 - Red Hat Enterprise Linux 4 Update 4 or Red Hat Enterprise Linux 5 on x86-64 (Intel(R) EM64T and AMD64
 - SUSE Linux Enterprise Server 9 or 10 on x86-64 (Intel(R) EM64T and AMD64
- ▶ Linux POWER™
 - Red Hat Enterprise Linux 5 on IBM eServer OpenPower™, System i™ or pSeries systems that support Linux
 - SUSE Linux Enterprise Server 10 on IBM eServer OpenPower, System i or pSeries systems that support Linux
- ▶ Solaris™
 - Solaris 9 and Solaris 10 64-bit kernel (UltraSPARC)

Windows

The Windows versions that support WLM are:

- ▶ Windows 2003 (x86 and x64)
- ▶ Windows XP Professional (x86 and x64)
- ▶ Windows Vista® (x86 and x64)

All systems based on Intel or AMD processors that are capable of running the supported Windows operating systems (x86, x64, and Itanium-based systems) are supported.

3.2 Planning DB2 environment

Plan your DB2 environment so that it is efficient, easily recoverable, and easy to maintain. The final solution depends mostly on your business needs. Planning an efficient and recoverable database environment is not an easy task and most of the planning topics are outside of the scope of this book. Here we give a few general guidelines.

Database environment

There's no special considerations for the database using WLM function. We provide a few good guidelines for setting up your database environment:

- ▶ Disk and file system setup:
 - Keep your data, active logs, archived logs, and instance home directories on separated file systems. These file systems need to rely on different sets of disks or LUNs when you use Storage Area Networks (SANs) to ensure that you are able to recover from disk failures.
 - Separate the operating system (OS) data on different sets of disks from database data disks to avoid OS and DB2 competing on the same I/O for disk resources.
 - Use external SAN disks whenever it is possible for easy maintenance and efficiency.
 - You need to have enough I/O to access your data. This means that you must have sufficient disk adapters, enough disks on your disk sets, and make sure data is evenly spread on the disks.
- ▶ Memory:
 - You must have enough memory for both the OS and the applications that run on the system. Size your database buffer pools to leave sufficient memory for the OS and possible disk caching. Oversizing buffer pools can cause OS swapping and slow down the OS and DB2.
 - If buffer pools over 1.7 GB on Linux and UNIX or 2 GB on Windows are required, you must have enough memory and 64-bit OS and DB2.

- ▶ CPU:
 - Consider symmetric multiprocessor (SMP) systems with two or more CPU for your database system. You are be able to run queries parallel on different CPUs to improve performance.

Application environment

To fully utilize DB2 WLM enhancements, plan how your applications use your databases so that you are able to differentiate different types of applications and work sets:

- ▶ Use different user accounts for different types of workloads when applicable.
- ▶ Do not use only one account to access all of your data.
- ▶ Do not use administrative accounts for applications.

A well-designed setup using different accounts for different applications allows you to grant the database privileges to each application based on the business requirement. This not only makes your database environment more secure, but also will make the WLM implementation much easier. You can better separate different types of workloads and manage them accordingly.

3.3 Lab environment

In this section we describe the lab environment built for writing this book.

3.3.1 Lab systems

We use following hardware to run our servers. AIX systems are the primary system used for the lab exercise.

- ▶ AIX servers
 - Clyde
 - pServer 690
 - 8 x 1.9GHz CPU
 - 8GB RAM
 - 2 x 72GB RAID 1 internal disks
 - 8 x 142GB RAID 0 +1 external disks
 - DB2 9.5 ESE
 - Bonnie
 - 4 x 1.9GHz CPU
 - 8GB RAM
 - 2 x 72GB RAID 1 internal disks

- 8 x 142GB RAID 0 + 1 external
- DB2 9.5 ESE
- ▶ Linux server
 - Puget
 - xSeries® 360
 - 2 x Intel(R) XEON(TM) MP CPU 2.00GHz
 - 2GB RAM
 - 2 x 72GB RAID 1 internal disks
 - DB2 9.5 ESE
 - ▶ Windows server
 - Cetus
 - ThinkCenter
 - Intel(R) Core(TM)2 CPU 6700 @ 2.66GHz
 - 4GB RAM
 - 2 x 72GB internal hard drives
 - DB2 9.5 ESE
 - Performance Expert 3.1
 - ▶ Storage
 - IBM DS4800
 - 2 x IBM EXP700
 - 16x IBM 143 GB

3.3.2 AIX server configuration

Figure 3-1 illustrates disk configuration for the partitioned database in the AIX servers. Operating system has its own volume group rootvg which is on internal disks. All database related file systems reside on the external SAN disks.

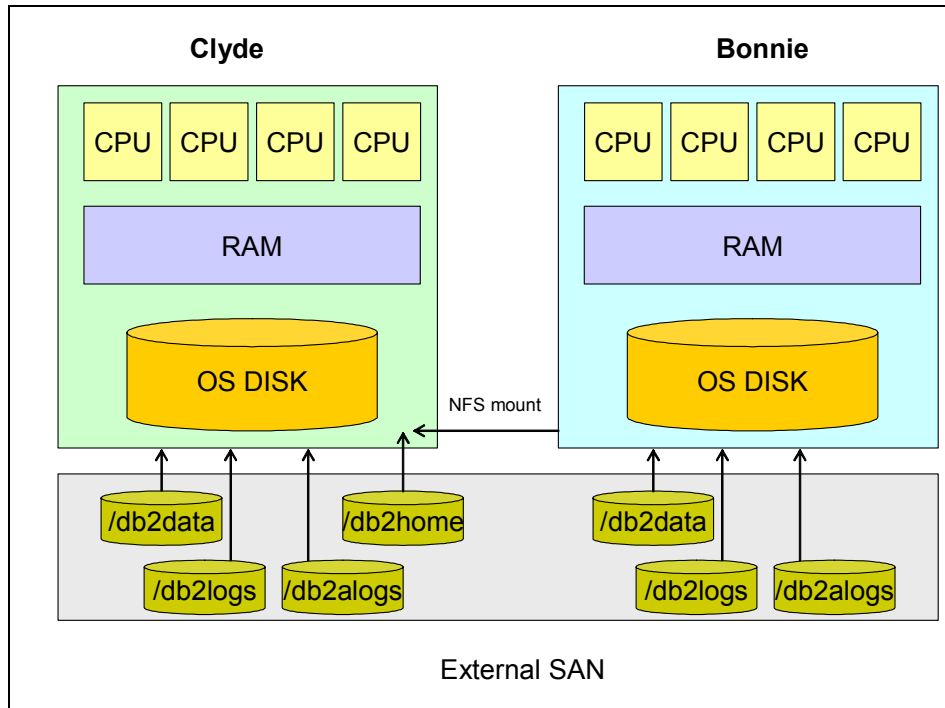


Figure 3-1 Disk setup

The instance home directory /db2home is on an external SAN disk attached to Clyde. This home directory on Clyde is configured as an exportable Network File System (NFS) file system for mounting the DB2 home directory over it. The table spaces, active logs, and archived logs for both Clyde and Bonnie have their own file systems on the external SAN disks:

- ▶ /db2data for table spaces and table space containers
- ▶ /db2logs for active transaction logs
- ▶ db2alogs for archived transaction logs

Note: This is a simple Lab environment that has only one LUN configured for all the database related file systems on SAN. On production environments you should have the file systems on their own LUNs.

Figure 3-2 illustrates the partition configuration of WLMDb database we created on the AIX systems. WLMDb spreads over two physical nodes and consists of five database partitions. Partition 0 on Clyde acts as coordinating partition. WLMDb has six database partition groups:

- ▶ IBMCATGROUP for system catalogs on partition0, the coordinating partition.

- ▶ IBMDEFAULTGROUP which spans over all five partitions. This has only one table space, USERSPACE1 which is the default table space for new tables. Our intention is not to use this table space for our tests.
- ▶ IBMTEMPGROUP which spans over all partitions. This has only one table space which is for temporary tables.
- ▶ NG1 is in partition 0 only. NG1 contains one table space for small referencing tables.
- ▶ NG2 spreads over database partition 1 to partition 4. This partition group has table spaces for our data for tests and benchmarking.
- ▶ NGALL spans over all database partitions. It has only one table space MAINT for event monitor data to be used for historical analyses.

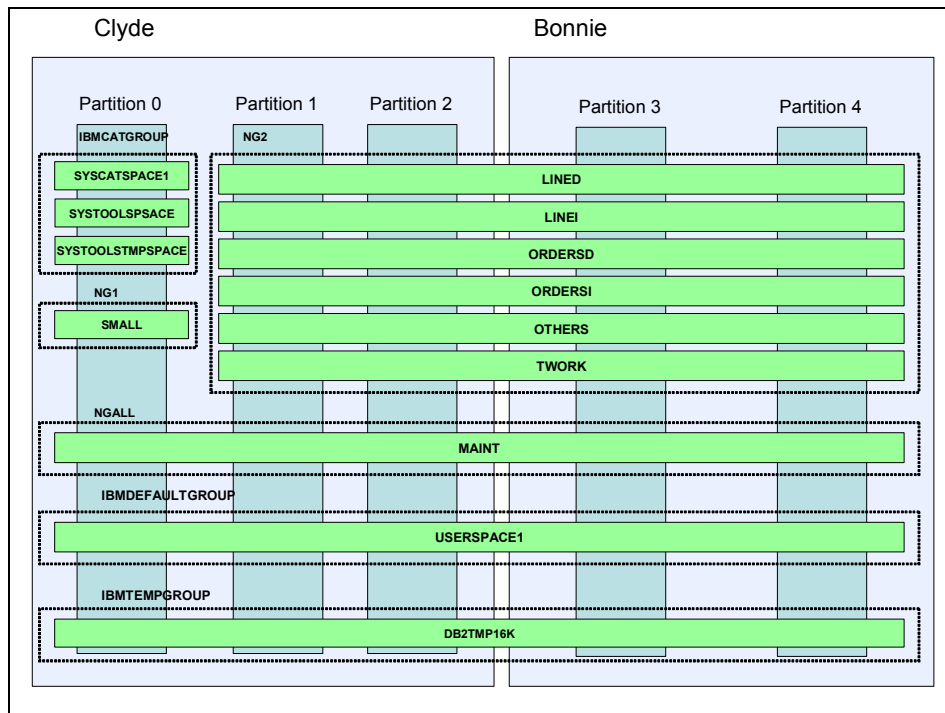


Figure 3-2 WLMDB partition configuration

3.3.3 TPC-H

We deploy TPC-H data for our test database. You can find more information about TPC-H at

<http://www.tpc.org/tpch/>

3.4 Installing DB2

The base function of DB2 9.5 Workload Manager is available in the core DB2 engine and the more advanced custom features are licensed under the Performance Optimization feature.

In this section, we guide you through the installation using the partitioned database design for our lab. The installation process is also applicable for the non-partitioned environment except the special steps we noted for a partitioned environment.

Installation on Linux and UNIX

In this section we go through DB2 9.5 ESE installation steps for AIX and Linux. You must have at least three user accounts and user groups for DB2:

- ▶ Instance owner: In our examples, we use `db2inst1`, which belongs to the group `db2adm`, as our instance owner.
- ▶ Fenced user: We use the user account `db2fenc1`, which belongs to the group `fencgrp`, for fenced processes.
- ▶ DB2 Administration Server account: We use `dasusr1`, which belongs to the group `dasgrp`.

If you install DB2 using `db2setup` command on the product CD, the installation program creates user accounts as well as the DB2 instance you specified during the installation.

If you use `db2_install` script on the product CD, you must set up user groups and DB2 accounts on your own. This is the approach we used to set up our DB2 environment.

In DB2 9.5, you can install DB2 as non-root user on Linux and UNIX platforms. In our lab, we continue use root to perform DB2 installation. The following lists the high-level steps we used to install DB2 on AIX and Linux environment:

1. Log in as root, or gain root access by issuing the following command:

```
su -
```
2. Create user accounts and groups as described above using operating system tools.
3. Mount the product CD:

```
On AIX: mount -V cdrfs -o ro /dev/cd0 /mnt  
On Linux mount /dev/cdrom /mnt
```
4. Change the working directory to `/mnt/ese`:

```
cd /mnt/ese
```

5. Start product installation:

```
./db2_install
```

6. Install DB2 at default location by answering *no* to the installing location question.
7. We select ESE as the product to be installed.
8. Once the product is installed, change the working directory to DB2 9.5 ESE instance directory:

```
On AIX: cd /opt/IBM/db2/V9.5/instance
```

```
On Linux: cd /opt/ibm/db2/V9.5/instance
```

9. Create an instance:

```
./db2icrt -u db2fenc1 db2inst1
```

This time we do not create DB2 Administration Server, which is usually one of the installation steps. It can be created later.

Post installation

Once DB2 is installed, you may need to modify some of the system files:

- ▶ Update Services file

Instance creation process automatically creates service entries in `/etc/services` file as shown in Example 3-1.

Example 3-1 Services file

```
DB2_db2inst1    60000/tcp
DB2_db2inst1    60000/tcp
DB2_db2inst1_1  60001/tcp
DB2_db2inst1_2  60002/tcp
DB2_db2inst1_END    60003/tcp
```

We modify the Services file to run our DB2 instance on listening port 50001 for the incoming connections. We also reserve ports 60000 - 60004 to be used in the partitioned environment. If you are not going to set up partitioned database, no change is required. You can then use `DB2_db2inst1` as your `SVCENAME`.

Example 3-2 shows the modified services file.

Example 3-2 Modified services file

```
db2inst1c      50001/tcp
DB2_db2inst1    60000/tcp
DB2_db2inst1_1  60001/tcp
```

```
DB2_db2inst1_2 60002/tcp
DB2_db2inst1_3 60003/tcp
DB2_db2inst1_4 60004/tcp
DB2_db2inst1_END      60005/tcp
```

► Setup communication

After creating DB2 instance, we configure DB2 communication:

- a. Log on as db2inst1 user or gain access by running su command:

```
su - db2inst1
```

- b. Update DB2 registry value DB2COMM:

```
db2set DB2COMM=TCPIP
```

- c. Update database manager configuration parameter TCP/IP service name:

```
db2 update dbm cfg using SVCENAME db2inst1c
```

► Update db2nodes.cfg

To setup the partitioned database as described in 3.3.2, “AIX server configuration” on page 46, DB2 is installed on both Clyde and Bonnie. Next, we modify db2nodes.cfg file under \$DB2HOME/sqllib/ directory to have five database partitions, three on Clyde and two on Bonnie. If you are not going run your database on partitioned environment, you can skip next step. Example 3-3 shows our db2nodes.cfg file.

Example 3-3 db2nodes.cfg

```
0 Clyde.itsosj.sanjose.ibm.com 0
1 Clyde.itsosj.sanjose.ibm.com 1
2 Clyde.itsosj.sanjose.ibm.com 2
3 Bonnie.itsosj.sanjose.ibm.com 0
4 Bonnie.itsosj.sanjose.ibm.com 1
```

► Set up ssh

There are two additional optional steps left for partitioned environment. We want our instance owner to be able to access from Clyde to Bonnie without prompted passwords and other way around. You can implement this either by using **rsh** or by using **ssh**. We used the later one. Our AIX servers already has ssh clients and servers installed. Since our instance owner db2inst1 has same home directory on both Bonnie and Clyde, we only have to create public and private ssh keys on one server. Example 3-4 shows how to create ssh keys:

Example 3-4 How to create ssh keys

```
>ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.  
Enter file in which to save the key (/db2home/db2inst1/.ssh/id_rsa):  
Created directory '/db2home/db2inst1/.ssh'.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /db2home/db2inst1/.ssh/id_rsa.  
Your public key has been saved in /db2home/db2inst1/.ssh/id_rsa.pub
```

After the keys are create, we log in once from Bonnie to Clyde and once from Clyde to Bonnie using ssh to ensure the setup is working. On the first time logging in with ssh from one server to another server, you will be prompted to accept servers public key to your *known_hosts* file. You must answer “yes” to this question. This step sets the required responses for the ssh prompting questions in automated connection. Without it, the automated connection from one host to another will fail. By doing the test, we also update the *know_hosts* file on both servers for later use.

Now we are ready to start our DB2 9.5 ESE by issuing the following command as user db2inst1:

```
db2start
```

Installation on Windows

You must install DB2 9.5 using an account that belongs to a local administrators or a domain administrators if you plan to use domain accounts for DB2 accounts. Launch *setup.exe* from your product CD. The installation setup guides you through the installation.

You can check Windows installation instructions from:

<https://publib.boulder.ibm.com/infocenter/db21uw/v9r5/topic/com.ibm.db2.luw.qb.server.doc/doc/t0052773.html>

Verify your installation

You can check if the DB2 installed indeed has WLM feature by creating a database and verify if one of the new WLM catalog tables is in place. Example 3-5 shows that we created a database WLMDB and select WLM catalog table SYSCAT.WORKLOADS. Two default workloads SYSDEFAULTUSERWORKLOAD and SYSDEFAULTADMWORKLOAD were created.

Example 3-5 Verifying your database is WLM capable

```
db2 create db WLMDB  
DB20000I The CREATE DATABASE command completed successfully.
```

```
db2 connect to WLMDB
```

connect to WLMDB

Database Connection Information

Database server = DB2/AIX64 9.5.0
 SQL authorization ID = DB2INST1
 Local database alias = WLMDB

db2 "SELECT WORKLOADID, SUBSTR(WORKLOADNAME,1,24) as WORKLOADNAME FROM SYSCAT.WORKLOADS"

WORKLOADID	WORKLOADNAME
1	SYSDEFAULTUSERWORKLOAD
2	SYSDEFAULTADMWORKLOAD

2 record(s) selected.

3.5 First steps

In this section we describe the default WLM configuration and what you can do with it.

3.5.1 The default DB2 WLM configuration

In DB2 9.5, every DB2 work is executed within a service class. Each user connection to database is mapped to a workload. DB2 creates two default workloads and three default service classes on the database you create. The default service classes are:

- ▶ SYSDEFAULTUSERCLASS
- ▶ SYSDEFAULTMAINTENANCECLASS
- ▶ SYSDEFAULTSYSTEMCLASS

Each default service superclass has one default service subclass SYSDEFAULTSUBCLASS.

The default workloads are:

- ▶ SYSDEFAULTUSERWORKLOAD
- ▶ SYSDEFAULTADMWORKLOAD

In Figure 3-3 illustrates the relationships between default workload and service classes.

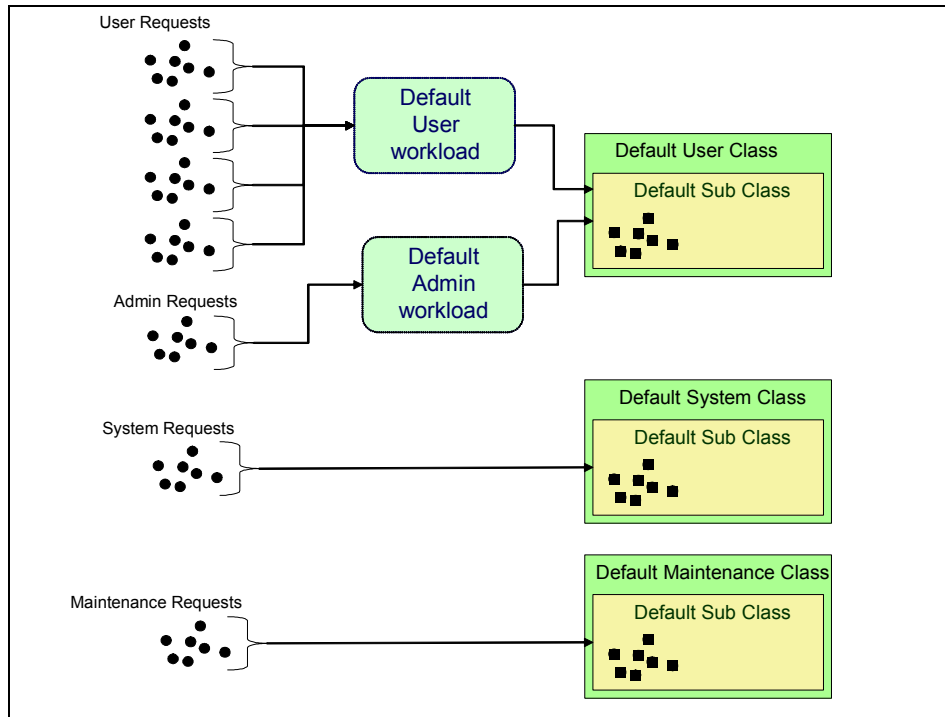


Figure 3-3 Default workloads and service classes

Without any customization, all the user requests are assigned to the default user workload `SYSDEFAULTUSERWORKLOAD` and executed under the default superclass `SYSDEFAULTUSERCLASS`. You are not able to disable, drop, define thresholds or create your own subclasses under `SYSDEFAULTUSERCLASS`. The only alter you can do on `SYSDEFAULTUSERCLASS` is to define `OUTBOUND CORRELATOR`, which is required if you want to integrate DB2 WLM with operating system workload manager. This special case is discussed in detail in Chapter 8, “AIX Workload Manager considerations” on page 193.

The default administration workload, `SYSDEFAULTADMWORKLOAD`, is a special DB2-supplied workload definition that is always mapped to `SYSDEFAULTUSERCLASS`. This workload is intended for the database administrator to perform their work or to take corrective actions when needed. As this workload is not affected by thresholds, it has limited workload management control and is not recommended for use in submitting regular day-to-day work.

All internal DB2 maintenance and administrative activities are executed on the default system maintenance class `SYSDEFAULTMAINTENANCECLASS`. Examples of internal maintenance and administrative tasks are:

- ▶ DB2 asynchronous background processing (ABP) connections
- ▶ Health monitor initiated backup
- ▶ Health monitor initiated RUNSTATS
- ▶ Health monitor initiated REORG

All the system requests are executed under the default system service class `SYSDEFAULTSYSTEMCLASS` on its default subclass. The default system service superclass is the execution environment for all internal connections and threads that perform system-level tasks. Examples of these tasks are:

- ▶ ABP daemon
- ▶ Connections issued by the Query Controller component of Query Patroller
- ▶ Self Tuning Memory Manager (STMM)
- ▶ Prefetcher engine dispatchable units (EDUs) (`db2pfchr`)
- ▶ Page cleaner EDUs (`db2pclnr`)
- ▶ Log reader EDUs (`db2loggr`)
- ▶ Log writer EDUs (`db2loggw`)
- ▶ Log file reader EDUs (`db2lfr`)
- ▶ Deadlock detector EDUs (`db2dlock`)
- ▶ Event monitors (`db2evm`)
- ▶ Connections performing system level tasks

Both `SYSDEFAULTMAINTENANCECLASS` and `SYSDEFAULTSYSTEMCLASS` are special service superclasses for DB2 internal use only. You cannot create subclasses under these service classes. You also cannot associate any workloads or work actions with them. Furthermore, you cannot implement thresholds on these special service classes.

On a default configuration, `SYSCAT.SERVICECLASSES` and `SYSCAT.WORKLOADS` are the only two tables that have content.

3.5.2 Monitoring the default WLM environment

The default WLM environment is a good starting point to understand the activities take place at the database. Even with the basic WLM environment, there are a lot of information available. You can use the new WLM table functions to collect statistical information for analysis or monitor your database activities real time.

In this section, we show you a few examples of using WLM table functions to see the default WLM settings and the activities in the DB2. The detail description of the WLM table functions is provided in Chapter 5, “Monitoring” on page 83.

Collecting statistical information

Using the statistical WLM table functions, you can collect the statistical information about the system for problem solving and analysis.

If you want see high watermark for concurrent connection since last statistics reset you can use table function WLM_GET_SERVICE_SUPERCLASS_STATS as shown in Example 3-6.

Example 3-6 High watermark of concurrent connections

```
SELECT SUBSTR(Service_superclass_name, 1, 26) AS service_superclass_name,
       LAST_RESET,
       CONCURRENT_CONNECTION_TOP CONCURRENT_CONN_TOP
FROM TABLE(WLM_GET_SERVICE_SUPERCLASS_STATS(' ', -2)) AS scstats;
```

SERVICE_SUPERCLASS_NAME	LAST_RESET	CONN_TOP
SYSDEFAULTSYSTEMCLASS	2007-08-28-10.12.30.192053	4
SYSDEFAULTMAINTENANCECLASS	2007-08-28-10.12.30.192076	1
SYSDEFAULTUSERCLASS	2007-08-28-10.12.30.192093	2

3 record(s) selected.

Example above shows high watermark connections for default service classes. Value of column “LAST_RESET” points to the time when the statistics were reset. How to reset statistics will be discussed in chapter 5.

If you are having performance problems with your applications connecting to your database, and you do not know whether problems are related to database or applications, you probably want to take closer look to your database first. You can use WLM table functions to further investigate the problem. In Example 3-7 we use table function WLM_GET_SERVICE_SUBCLASS_STATS to collect the following statistics data:

- ▶ The number of actions that are completed after last reset.
- ▶ The high watermark for concurrent connections.
- ▶ The average action lifetime.

Example 3-7 Using table functions to investigate performance

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS superclass_name,
       SUBSTR(CHAR(SUM(COORD_ACT_COMPLETED_TOTAL)),1,13) AS actscompleted,
       SUBSTR(CHAR(MAX(CONCURRENT_ACT_TOP)),1,6) AS actshw,
       CAST(CASE WHEN SUM(COORD_ACT_COMPLETED_TOTAL) = 0 THEN 0
             ELSE SUM(COORD_ACT_COMPLETED_TOTAL * COORD_ACT_LIFETIME_AVG)
                / SUM(COORD_ACT_COMPLETED_TOTAL) END / 1000 AS DECIMAL(9,3))
       AS actavglifetime
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('SYSDEFAULTUSERCLASS', ' ', -2)) AS
scstats
GROUP BY SERVICE_SUPERCLASS_NAME
ORDER BY SUPERCLASS_NAME;
```


SUPERCLASS_NAME	ACTSCOMPLETED	ACTSHW	ACTAVGLIFETIME
SYSDEFAULTUSERCLASS	2088	31	1.348

1 record(s) selected.

This information can help us understand if the database is performed to our expectation, if database configuration change is required, or if it is necessary to take closer look at applications and queries.

If you want to look at the statistics from workload point of view, you can use table function `WLM_GET_WORKLOAD_STATS` to see what is the high watermark for concurrent workloads.

Example 3-8 shows how to get data for default workloads using table function `WLM_GET_WORKLOAD_STATS`. We see that the highest number of concurrent occurrences is two in `SYSDEFAULTUSERWORKLOAD`. Highest number of concurrent activities is one in `SYSDEFAULTUSERWORKLOAD`. No activities have been mapped to `SYSDEFAULTADMWORKLOAD` at this time.

Example 3-8 Workload table function

```
SELECT SUBSTR(WORKLOAD_NAME,1,22) AS w1_def_name,
       CONCURRENT_WLO_TOP AS wlo_top,
       CONCURRENT_WLO_ACT_TOP AS wlo_act_top
FROM TABLE(WLM_GET_WORKLOAD_STATS(CAST(NULL AS VARCHAR(128)), -2))
AS w1stats
ORDER BY w1_def_name;
```

WL_DEF_NAME	WLO_TOP	WLO_ACT_TOP
SYSDEFAULTADMWORKLOAD	0	0
SYSDEFAULTUSERWORKLOAD	2	1

2 record(s) selected.

WLM real-time monitoring table functions

In the previous section, we show you how to use statistical table functions to check the database performance. If you want to see what is happening in your database just now, you also can WLM table functions to view runtime information.

You can use `GET_SERVICECLASS_WORKLOAD_OCCURENCES` table function to see which workload occurrences are currently running on the system as shown in Example 3-9.

Example 3-9 Workload occurrences table function

```

SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS superclass_name,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS subclass_name,
       SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS apphndl,
       SUBSTR(WORKLOAD_NAME,1,18) AS workload_name,
       SUBSTR(CHAR(WORKLOAD_OCCURRENCE_ID),1,6) AS wlo_id
FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES
            (CAST(NULL AS VARCHAR(128)), CAST(NULL AS VARCHAR(128)), -2));

```

SUPERCLASS_NAME	SUBCLASS_NAME	APPHNDL	WORKLOAD_NAME
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	174	SYSDEFAULTUSERWORK
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	207	SYSDEFAULTUSERWORK

2 record(s) selected.

In Example 3-10 we show how to list agents executing on a service class using table function WLM_GET_SERVICE_CLASS_AGENTS.

Example 3-10 How to find out how many agents are running

```

SELECT SUBSTR(AGENTS.SERVICE_SUPERCLASS_NAME,1,19) AS superclass_name,
       SUBSTR(AGENTS.SERVICE_SUBCLASS_NAME,1,19) AS subclass_name,
       COUNT(*) AS AGENT_COUNT
FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('', '', CAST(NULL AS BIGINT), -2)) AS
agents
WHERE agent_state = 'ACTIVE'
GROUP BY service_superclass_name, service_subclass_name
ORDER BY service_superclass_name, service_subclass_name;

```

SUPERCLASS_NAME	SUBCLASS_NAME	AGENT_COUNT
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	4
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	1

2 record(s) selected.

If you wish to find out all the activities for a specific application, you can use table function WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES. This table function accepts two input parameters: application handle and partition number. First, you need to find out the application handle for the application using **db2 list applications** command. We used -1 for partition number, which points to current partition. Example 3-11 shows all activities for application with application handle 384.

Example 3-11 Finding workload occurrence activities

```

SELECT SUBSTR(CHAR(UOW_ID),1,5) AS UOWID,
       SUBSTR(CHAR(ACTIVITY_ID),1,5) AS ACTID,
       SUBSTR(CHAR(PARENT_UOW_ID),1,8) AS PARUOWID,
       SUBSTR(CHAR(PARENT_ACTIVITY_ID),1,8) AS PARACTID,
       ACTIVITY_TYPE AS ACTTYPE,
       SUBSTR(CHAR(NESTING_LEVEL),1,7) AS NESTING
FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(384, -1)) AS WLOACTS
ORDER BY UOWID, ACTID;

```

UOWID	ACTID	PARUOWID	PARACTID	ACTTYPE	NESTING
3	1	-	-	READ_DML	0

1 record(s) selected.

If you wish to find detailed information about certain application, you can use table function `WLM_GET_ACTIVITY_DETAILS`. This table function returns detailed information about the specified activity. See Example 3-12.

Example 3-12 Collecting activity information with table function

```

SELECT SUBSTR(NAME, 1, 20) AS name,
       SUBSTR(VALUE, 1, 30) AS value
FROM TABLE(WLM_GET_ACTIVITY_DETAILS(386,1,1,-1)) AS acctdetail
WHERE NAME IN ('APPLICATION_HANDLE',
              'LOCAL_START_TIME',
              'UOW_ID',
              'ACTIVITY_ID',
              'PARENT_UOW_ID',
              'PARENT_ACTIVITY_ID',
              'ACTIVITY_TYPE',
              'NESTING_LEVEL',
              'INVOCATION_ID',
              'ROUTINE_ID');

```

NAME	VALUE
APPLICATION_HANDLE	386
UOW_ID	1
ACTIVITY_ID	1
PARENT_UOW_ID	
PARENT_ACTIVITY_ID	
ACTIVITY_TYPE	READ_DML
NESTING_LEVEL	0
INVOCATION_ID	0
ROUTINE_ID	0

LOCAL_START_TIME 2007-08-28-20.11.30.343467

4



Customizing the WLM execution environments

Now that we have worked through the default WLM setup, it is now time begin building a WLM plan and implementing the plan. From this plan we can evolve our WLM implementation. In this chapter we describe the steps for customizing the DB2 WLM for achieving business objectives. We discuss the methodology for building and evolving a WLM implementation.

4.1 Stages of workload management

Typically a DB2 database system has several different kinds of workloads, each with their own resource and availability requirements. They often undergo similar changes throughout their life cycle. For example, Sales reporting may require large reports on a monthly or quarterly basis. Where as, Inventory may require extensive re-casting of inventory quarterly or annually. These requirements may change over time as new business are acquired or applications are merged into the database system or the data in the database system simply grows. The keys to managing these workloads and protecting their needed resources are:

- ▶ Identify the work

The management goals are either given or can be derive from business objectives, service delivery, or performance objective. To achieve a goal, you first must be able to identify details about the work. There are many different sources that you can use to identify database activities, for example, user ID, user group, application name, session name and so on. Activities can be of varying types: system activities, administration activities, utilities, and applications. You can group the activities based on business area, activity type, or service requirements for DB2 WLM to manage.

- ▶ Manage the work

During this stage, you determine how you want to manage the work and current resources to meet your goals. The generic way to do this is to assign resources based on the characteristics of work, create and impose controls.

Database work can be managed based on request or resource:

- Request management

In the request management, work is managed based on its priority that are critical to business. Similar type of work are grouped together and set threshold and actions when request exceed predefined conditions

- Resource management

In resource management, similar type of work are grouped based on same business priority and control is placed to limit excessive and unexpected resource consumption.

- ▶ Monitor the work

Monitoring is for determining whether you are achieving a goal and identifying the problems that might be preventing you from achieving your goal. You can capture the activity information, store them, and analyze the data.

This is a continuous cycle that can be expressed as a cyclical process as shown in Figure 4-1.

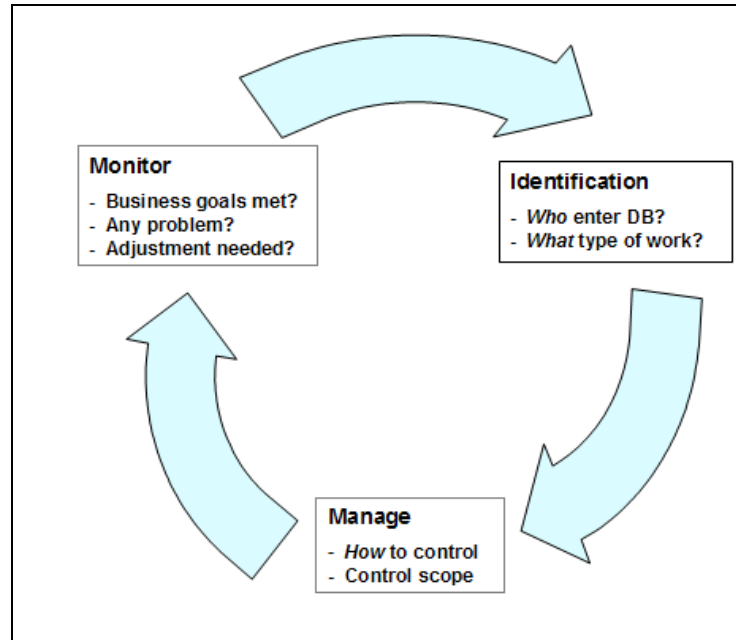


Figure 4-1 WLM methodology cycle

4.2 Identify the work

The process of identifying workloads can be very complex. It is best to start by using what you already know. Many key facts about the workload is often stated from the very beginning and based on business requirements:

- ▶ What are the business requirements?
- ▶ Are there any Service Level Agreements (SLA) that must be met?
- ▶ Are there any management or user requirements?

Develop a comprehensive view of works in the database that need to meet these business requirements. Next, identify the characteristics and changes to the workload by using workload profiling.

Checklist

In the process of identifying the workload, trying to gather as much as information about the workload. In the management stage, these information will be used to customize the WLM to manage your workload to meet the business goal. The following lists what should be identified:

- ▶ **Tasks:**

What processes do you want to identify or what you have discovered from previous monitoring? For example utilities run by DBA, all the report jobs and so on.
- ▶ **Business requirements:**

What is the purpose or function of the task? What requirements are attached to this task. These requirements should be expressed in terms of desired outcome, such as an SLA or business restrictions.

For the workload we identified, we
- ▶ **Task identity:**

How is the process identified in the system? This is for WLM to identify what workload comes to the database system. WLM can identify a process or workload using the following:

 - Application name (APPLNAME)
 - Authorization ID (SYSTEM_USER)
 - SESSION_USER
 - SESSION_USER GROUP
 - SESSION_USER ROLE
 - CLIENT_USERID
 - CLIENT_APPLNAME
 - CLIENT_WRKSTANNAME
 - CLIENT_ACCTNG

For each task, you must identify at least one of these for WLM.
- ▶ **Action:**

What action is needed for this process? In other words, what do you want to happen to control or measure this task. WLM provides the following actions:

 - Database level data collection:
 - High level data collection:

The aggregate activity data will be captured for the service class and sent to the applicable event monitor. The information is collected periodically on an interval that is specified by the `wlm_collect_int` database configuration parameter. The SQL statement clauses are

```
COLLECT AGGREGATE REQUEST DATA  
COLLECT AGGREGATE ACTIVITY DATA [BASE | EXTENDED]
```
 - Detailed data collection (at all levels: database, service class, workload:

The information about each activity that executes in the service class will be sent to the applicable event monitor when the activity completes. The SQL statement clause is

```
COLLECT ACTIVITY DATA
```

- Controls on processes:

Specify what to control and the control scope use WORK ACTION or THRESHOLD.

4.2.1 Workload identify worksheet

Use a worksheet to list the tasks, business requirements, process identify, and action identified.

The workload example we are using is a typical starting point for many data warehouse customers. We broke down our workload as follows:

- ▶ Administrative tasks:
We categorized the work such as security, monitoring, data maintenance (backup, recovery, REORG, RUNSTATS) as administrative tasks. The task identification is the group ID DB2ADM.
- ▶ Batch work
The batch work we identified includes loading data from various external sources and Extract Transform Load applications (ETL). The workload identification is etl.exe or client user ID BATCH.
- ▶ Production Ad hoc queries
These may come from a vendor tool or from end users. The users in this category are all assigned to the group DSSGROUP.
- ▶ Production reports
These are generated reports built using a vendor tool such as Brio, Cognos, Microstrategy, etc. They are identified by their executable name. In our example, they run dss.exe.

For these workload we identified, we want to start by knowing what resources do they consume, how often do they run, when do they run. These information can help us decide if we need to take any action to protect this workload and what actions do we need to take to limit the workloads impact on other workloads.

Table 4-1 shows our starting point. Remember, workload management is a continuous cycle of identification, management, and monitoring. This is merely a starting point of identification. Once we have gained more information and knowledge about our workloads, we can develop a more comprehensive plan.

Table 4-1 Workload identify worksheet

Task	Business requirements	Identification	Action
Admin	Manages the database environment	groupid = DB2ADM	Report the times, duration, and frequency of tasks
Batch	ETL must be complete prior to primetime shift	Loads and other ETL process using etl.exe or client userid = 'BATCH' and utility LOAD	report the times, duration, and frequency of tasks
Production	Prime time 8:00 AM to 6:00 PM	Ad Hoc queries and reports run under dss.exe	Identify ad hoc separately from reports
_Ad hoc queries	Must complete 90% < 5 minutes	groupid = dssgroup	Report the times, duration, and frequency of tasks
_Analysis Reports	Must complete all reports daily	exec = dss.exe	Report the times, duration, and frequency of tasks

4.3 Manage the work

The WLM management stage is to build steady progress to meet the business goal and actions that you can take when there are indications that the goal is not being met. Use the worksheet prepared to build the DB2 WLM objects:

- ▶ Service class
To collect historical analysis reports for each task, we create service subclasses for each task.
- ▶ Workload
To assign the task activities for each service classes, we create workloads using Identification.
- ▶ Event monitor
To collect and store aggregate information in tables, we create a write-to-table statistics event monitor.

In our basic setup, we want to historical analysis reports.

4.3.1 Creating the service classes

The service classes give us a hierarchy for assigning work. They are the basic building blocks for all workloads. Use the CREATE SERVICE CLASS statement to define the service class. If you want to collect statistics data, specify the COLLECT AGGREGATE ACTIVITY clause.

Figure 4-2 illustrate the service classes we defined. The service classes address the WLM worksheet column *Action*. We could have altered the default service classes and workloads but we want to establish a foundation to evolve our WLM setup. Our superclass is HIGHLVL under which all of our subclasses are assigned. Our subclasses are used to describe what action we want to take for all workloads assigned to the particular subclass. Subclasses allow us to be very specific about what action is performed on specific workloads.

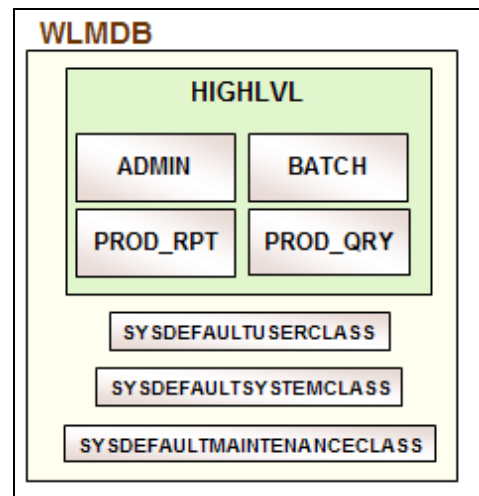


Figure 4-2 WLMDB service classes

Example 4-1 shows service classes DML. We have chosen to use aggregate levels of collection, COLLECT AGGREGATE ACTIVITY DATA BASE. This level of data collection has the least impact on the system and presents a high level of information in our monitoring. Additionally, we want both aggregate activity and aggregate request data.

Example 4-1 Creating service classes for basic setup

```

CREATE SERVICE CLASS highlvl DISABLE;
CREATE SERVICE CLASS admins UNDER HIGHLVL COLLECT AGGREGATE ACTIVITY DATA BASE
DISABLE;
  
```

```
CREATE SERVICE CLASS batch UNDER HIGHLVL COLLECT AGGREGATE REQUEST DATA BASE
DISABLE;
CREATE SERVICE CLASS prod_rpt UNDER HIGHLVL COLLECT AGGREGATE ACTIVITY DATA
EXTENDED DISABLE;
CREATE SERVICE CLASS prod_qry UNDER HIGHLVL COLLECT AGGREGATE ACTIVITY DATA
EXTENDED DISABLE;
```

Note: Disable all service classes and workloads until you are ready to use them. This keeps work from being assigned a service class or workload prior to completing the WLM setup. In a script the timing window may be small but when using the Command Line, the exposure is much longer

4.3.2 Creating the workloads

Use CREATE WORKLOAD statement to define the workloads and specify which subclass is responsible for handling the workload. The workloads addresses the WLM worksheet column *Identification*. Each workload is tied to a subclass.

Figure 4-3 illustrates the sample workloads we defined in our database WLMDB.

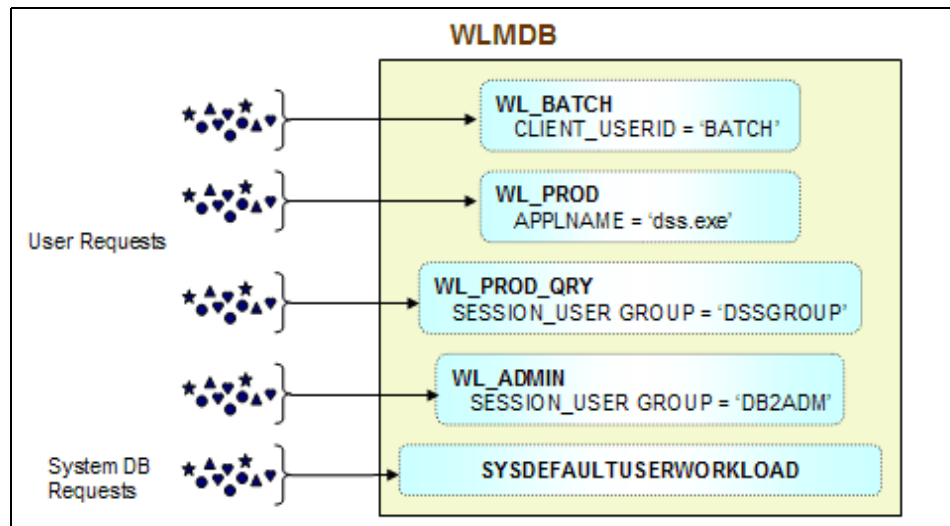


Figure 4-3 WLMDB workloads

Example 4-2 shows the workloads we set up, again, allowing the WLM setup to evolve.

Example 4-2 Creating workloads

```
CREATE WORKLOAD w1_batch CURRENT CLIENT_USERID ('BATCH')
```

```

DISABLE SERVICE CLASS BATCH UNDER highlvl POSITION AT 1;

CREATE WORKLOAD w1_prod_rpt APPLNAME ('dss.exe')
  DISABLE SERVICE CLASS prod_rpt UNDER highlvl POSITION AT 2;

CREATE WORKLOAD w1_prod_qry SESSION_USER GROUP ('DSSGROUP')
  DISABLE SERVICE CLASS prod_qry UNDER highlvl POSITION AT 3;

CREATE WORKLOAD w1_admin SESSION_USER GROUP ('DB2ADM')
  DISABLE SERVICE CLASS admins UNDER highlvl POSITION AT 4;

```

Note: The workloads should be sequenced to specify the order of workload assignment. Don't assume they are being assigned based on the order of creation. The default position when adding a workload is LAST. As the WLM setup evolves, it becomes increasingly difficult to keep them in order.

Set client information

The workload identification attributes are based either on the server identification or the client identification, as in three tier applications. The server identifications are:

- ▶ APPLNAME
- ▶ SYSTEM_USER
- ▶ SESSION_USER
- ▶ SESSION_USER GROUP
- ▶ SESSION_USER ROLE

The client identifications are:

- ▶ CLIENT_USERID
- ▶ CLIENT_APPLNAME
- ▶ CLIENT_WRKSTANNAME
- ▶ CLIENT_ACCTNG

Prior to DB2 9.5, these were set either using the client `db2cli.ini` or the set client information API (*sqleseti*). Starting in DB2 9.5, the client identification can also be set at the server. This adds flexibility to workload identification for work initiated on the server or in a three tier environment. In our example we want all batch jobs identified by the `client_userid` BATCH. Example 4-3 shows a batch job using the `wlm_set_client_info` stored procedure.

Example 4-3 Batch job using the wlm_set_client_info

```

db2 connect to w1mdb
db2 "call sysproc.wlm_set_client_info('BATCH',NULL,'MQT105',NULL,NULL)"
db2 -tvf /batchjobs/mqt/refresh_mqt_current_yr_sls.clp

```

db2 reset

Note: Using the call `sysproc.wlm_set_client_info` stored procedure on the server side, extends the flexibility of identifying work.

4.3.3 Allowing use of the WLM setup

Before the workloads can be used, permission must be granted. You can grant the USAGE privilege to specific users, groups, roles, or PUBLIC. In our example we granted all workloads to PUBLIC since there are no security or audit concerns in the Lab environment. See Example 4-4.

Example 4-4 Grant workload usage

```
GRANT USAGE ON WORKLOAD WL_ADMIN TO PUBLIC;  
GRANT USAGE ON WORKLOAD WL_BATCH TO PUBLIC;  
GRANT USAGE ON WORKLOAD WL_PROD_RPT TO PUBLIC;  
GRANT USAGE ON WORKLOAD WL_PROD_QRY TO PUBLIC;
```

Next the service classes and workloads are enabled. As stated earlier, we recommend to create all service classes and workloads using DISABLED to prevent premature usage until the entire WLM setup has been completed.

4.3.4 Creating the event monitor

The final step in our basic setup is to create the event monitor. Since we have chosen to collect aggregate statistics for historical analysis, an event monitor is needed. In Example 4-5 we decided to write the event monitor data to tables instead of files. Since the amount of data is expected to be small, we wanted the flexibility of tailoring our reports using SQL.

Example 4-5 Creating event monitor using tables

```
CREATE EVENT MONITOR basic_mon FOR STATISTICS WRITE TO TABLE  
  SCSTATS (TABLE scstats_basic_mon IN maint),  
  WCSTATS (TABLE wcstats_basic_mon IN maint),  
  QSTATS (TABLE qstats_basic_mon IN MAINT),  
  WLSTATS (TABLE wlstats_basic_mon IN maint),  
  HISTOGRAMBIN (TABLE histogrambin_basic_mon IN maint),  
  CONTROL (TABLE control_basic_mon IN maint)  
  AUTOSTART;  
SET EVENT MONITOR basic_mon STATE 1;
```

Example 4-6 shows the event monitor tables created. We appended the event monitor name to the table names to make them unique and to correlate them to the event monitor. We created all the tables even though we are only doing aggregate collection, which uses the tables:

- ▶ SCSTATS
- ▶ WLSTATS
- ▶ HISTOGRAMBIN

The other table are created in case we need them later. All the event monitor table are created in a specific table space as a good administration practice.

Example 4-6 Event monitor tables

```
->db2 list tables for user
```

Table/View	Schema	Type	Creation time
CONTROL_BASIC_MON	ADMINHM	T	2007-08-28-12.54.53.340071
HISTOGRAMBIN_BASIC_MON	ADMINHM	T	2007-08-28-12.54.55.518744
QSTATS_BASIC_MON	ADMINHM	T	2007-08-28-12.54.55.111208
SCSTATS_BASIC_MON	ADMINHM	T	2007-08-28-12.54.53.931222
WCSTATS_BASIC_MON	ADMINHM	T	2007-08-28-12.54.54.326150
WLSTATS_BASIC_MON	ADMINHM	T	2007-08-28-12.54.54.733324

4.3.5 Using SYSDEFAULTADMWORKLOAD

The default administration workload SYSDEFAULTADMWORKLOAD is a special DB2-supplied workload definition that is not subject to any DB2 thresholds. This workload is intended to allow the database administrator to perform their work or to take corrective actions, as required. As this workload is not affected by thresholds, it has limited workload management control and is not recommended for use in submitting regular day-to-day work.

You can use the SET WORKLOAD command to assign a connection to the SYSDEFAULTADMWORKLOAD as follows:

```
SET WORKLOAD TO SYSDEFAULTADMWORKLOAD;
```

If this is the first occurrence of creating a workload or if you are not redoing the entire WLM setup, this statement is not needed. However, if you have a script which is constructed to delete the prior WLM configuration, an SQL4714 error may occur when all WLM service classes are disabled. The reason being, if all service classes have been disabled, nothing else in the script can execute until the work is routed to an enabled workload. This is where SYSDEFAULTADMWORKLOAD comes to the rescue.

Example 4-7 shows a full script used in the test environment.

Example 4-7 Script to delete and rebuild WLM setup

```

-----
-- set all existing work to default workload
-----
SET WORKLOAD TO SYSDEFAULTADMWORKLOAD;
-----
-- create WLM environment
-----
CREATE SERVICE CLASS HIGHLVL DISABLE;
CREATE SERVICE CLASS ADMINS UNDER HIGHLVL COLLECT AGGREGATE ACTIVITY DATA BASE
DISABLE;
CREATE SERVICE CLASS BATCH UNDER HIGHLVL COLLECT AGGREGATE REQUEST DATA BASE
DISABLE;
CREATE SERVICE CLASS PROD_RPT UNDER HIGHLVL COLLECT AGGREGATE ACTIVITY DATA
EXTENDED DISABLE;
CREATE SERVICE CLASS PROD_QRY UNDER HIGHLVL COLLECT AGGREGATE ACTIVITY DATA
EXTENDED DISABLE;
-----
-- identify workloads and assign to service classes
-----
CREATE WORKLOAD WL_BATCH CURRENT CLIENT_USERID ('BATCH') DISABLE SERVICE CLASS
BATCH UNDER HIGHLVL POSITION AT 1;
CREATE WORKLOAD WL_PROD_RPT APPLNAME ('dss.exe') DISABLE SERVICE CLASS
PROD_RPT UNDER HIGHLVL POSITION AT 2;
CREATE WORKLOAD WL_PROD_QRY SESSION_USER GROUP ('DSSGROUP') DISABLE SERVICE
CLASS PROD_RPT UNDER HIGHLVL POSITION AT 3;
CREATE WORKLOAD WL_ADMIN SESSION_USER GROUP ('DB2ADM') DISABLE SERVICE CLASS
ADMINS UNDER HIGHLVL POSITION AT 4;
-----
-- grant usage of workloads
-----
GRANT USAGE ON WORKLOAD WL_ADMIN TO PUBLIC;
GRANT USAGE ON WORKLOAD WL_BATCH TO PUBLIC;
GRANT USAGE ON WORKLOAD WL_PROD_RPT TO PUBLIC;
GRANT USAGE ON WORKLOAD WL_PROD_QRY TO PUBLIC;
-----
-- Enable the service classes
-----
ALTER SERVICE CLASS HIGHLVL ENABLE;
ALTER SERVICE CLASS ADMINS UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS BATCH UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS PROD_RPT UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS PROD_QRY UNDER HIGHLVL ENABLE;

ALTER WORKLOAD WL_ADMIN ENABLE;

```



```

ALTER WORKLOAD WL_BATCH ENABLE;
ALTER WORKLOAD WL_PROD_RPT ENABLE;
ALTER WORKLOAD WL_PROD_QRY ENABLE;
COMMIT;
-----
-- start using the new WLM setup
-----
SET WORKLOAD TO AUTOMATIC;
-----
-- setup and turn on the event monitor
-----
CREATE EVENT MONITOR BASIC_MON FOR STATISTICS WRITE TO TABLE
  SCSTATS (TABLE SCSTATS_BASIC_MON IN MAINT),
  WCSTATS (TABLE WCSTATS_BASIC_MON IN MAINT),
  QSTATS (TABLE QSTATS_BASIC_MON IN MAINT),
  WLSTATS (TABLE WLSTATS_BASIC_MON IN MAINT),
  HISTOGRAMBIN (TABLE HISTOGRAMBIN_BASIC_MON IN MAINT),
  CONTROL (TABLE CONTROL_BASIC_MON IN MAINT)
  AUTOSTART;
SET EVENT MONITOR BASIC_MON STATE 1;

```

Without the `SET WORKLOAD TO SYSDEFAULTADMWORKLOAD` command, once all the service classes are disabled, the script can not continue because we disabled every service class we are using. An SQL471N appears when the command shown is attempted

```
CREATE SERVICE CLASS HIGHLVL DISABLE;
```

4.4 Monitor the work

The last stage in a workload management cycle is monitoring the activities, analyzing the collected data, and verifying if the customized WLM environment can manage and control the workload as planned. Using monitoring, we can address the WLM worksheet column *Business requirements*.

In our example, we collect aggregate data over time and monitor it periodically. From the reports, we hope to learn about our workloads, determine if additional action is needed. The details of monitoring are covered in Chapter 5, “Monitoring” on page 83. Here we discuss only the reports used in our basic WLM setup.

Since we setup to collect aggregate statistics (both request and activity), we have data in the following tables:

- ▶ SCSTATS_BASIC_MON
- ▶ WLSTATS_BASIC_MON

► HISTOGRANBIN_BASIC_MON

Each table gives us a different perspective of the WLM setup.

Looking at the SCSTATS_BASIC_MON, we can export and analyze our workload at the subclass level. Example 4-8 shows the export SQL statements.

Example 4-8 Exporting event monitor data

```
EXPORT TO /tmp/exports/scstats_all.csv OF DEL
SELECT
  DATE(statistics_timestamp) AS stat_date,
  TIME(statistics_timestamp) AS stat_time,
  SUBSTR(service_subclass_name,1,10) AS subclass_name,
  CASE WHEN 0 > INT(SUM(concurrent_act_top))
    THEN 0
    ELSE INT(SUM(concurrent_act_top))
  END AS con_act_top,
  CASE WHEN 0 > INT(SUM(concurrent_connection_top))
    THEN 0
    ELSE INT(SUM(concurrent_connection_top))
  END AS CON_CONN_TOP,
  CASE WHEN 0 > INT(SUM(coord_act_completed_total))
    THEN 0
    ELSE INT(SUM(coord_act_completed_total))
  END AS coord_act_comp,
  CASE WHEN 0 > INT(SUM(coord_act_exec_time_avg))
    THEN 0
    ELSE INT(SUM(coord_act_exec_time_avg)/1000)
  END AS avg_c_exe_tm,
  CASE WHEN 0 > INT(SUM(request_exec_time_avg))
    THEN 0
    ELSE INT(SUM(request_exec_time_avg)/1000)
  END AS avg_r_exe_tm
FROM scstats_basic_mon
WHERE CHAR(DATE(statistics_timestamp)) = CURRENT DATE
GROUP BY DATE(statistics_timestamp), TIME(statistics_timestamp),
  SUBSTR(service_subclass_name,1,10)
ORDER BY 1,2,3
```

From the SCSTATS_Basic_MON table, for each time period, we can analyze:

- Top concurrent activity
- Top concurrent connections
- Top coordinator activity
- Coordinator activity completed
- Coordinator execution time (microseconds)
- Request execution time (microseconds)

Figure 4-4 shows the requests execution time by subclass of a typical day.

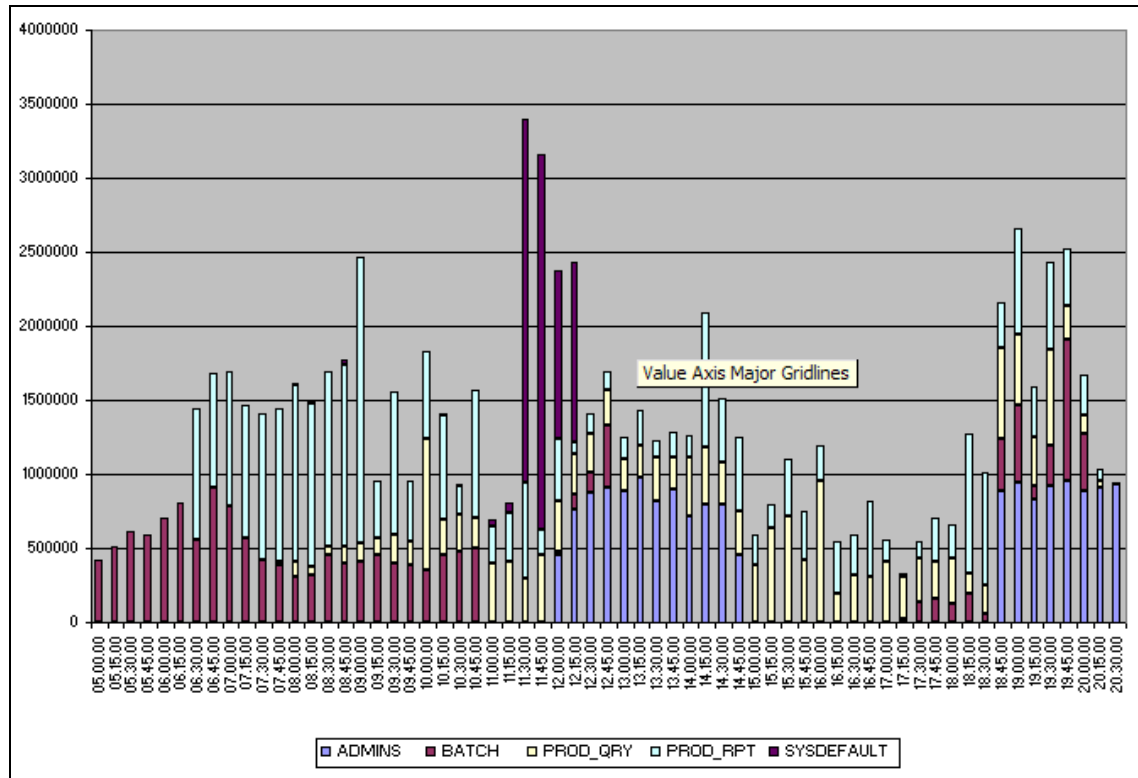


Figure 4-4 Request execution time by subclass

From this graph we see several interesting observations in a quick and easy to identify format:

- ▶ Our BATCH processing is from 5:00 AM to 11:00 AM with additional batch process running between 12:00 PM and 1:00 PM.
- ▶ Production reports begin at 6:30 AM and continue all during the prime shift and concludes at 8:00 PM because we have users in several time zones running reports. A heavy CPU demand is placed on our data warehouse during 8:00 AM and 9:00 AM.
- ▶ Production ad-hoc queries begin at 8:00 AM and run steady until 8:00 PM. Again we have users in several timezones. We also see heavy CPU loads at 9:00 AM and again around 4:00 PM.
- ▶ The Admins appear to be running heavy loads twice a day from noon to 1:00 PM and again from 6:00 PM until 8:30 PM. From our inquiry, we discover that the online table space backups are run during these times daily.

- ▶ We also see and unaccounted for workload in the SYSDEFAULTSUBCLASS. Some one is placing a very heavy load on the system around 11:30 AM. More investigation is needed to determine what is causing the additional load.

Using the same information but now formatted as a stacked bar graph, We see the accumulated affect of our workloads on the system as shown in Figure 4-5. This graph gives us the total perspective of how the workloads impact our system CPU consumption. Using this information, workloads can be rescheduled, prioritized, or limited using queues if they consume too much CPU at the same time.

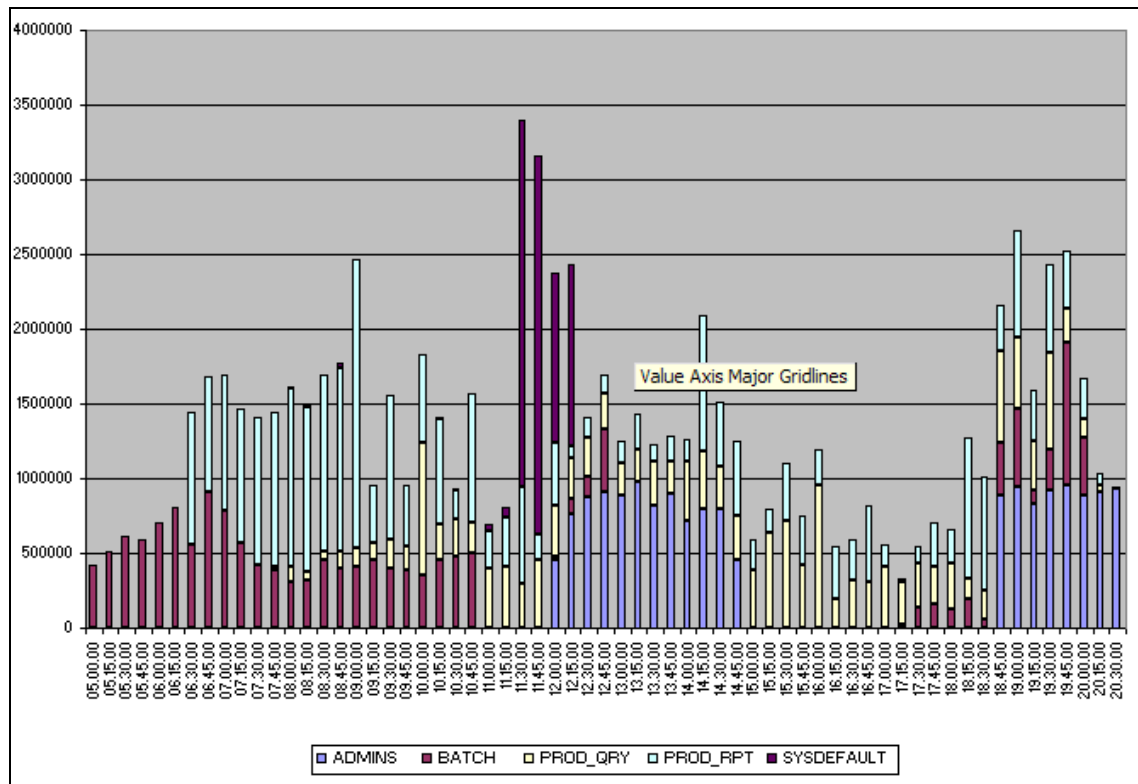


Figure 4-5 Bar chart - Requests execution time by subclass

Turning our attention to the active connections, we get a sense of how many workloads are running during the day as shown in Figure 4-6 on page 77. We see from this graph that our production ad-hoc queries usually peak twice a day. Our batch jobs appear to taper off around 11AM. We also see activity in the SYSTEMDEFAULTSUBCLASS during the prime shift between 11:00 AM and 1:30 PM. As shown in Figure 4-4 on page 75, this has sever impact on our CPU availability. More investigation is needed.

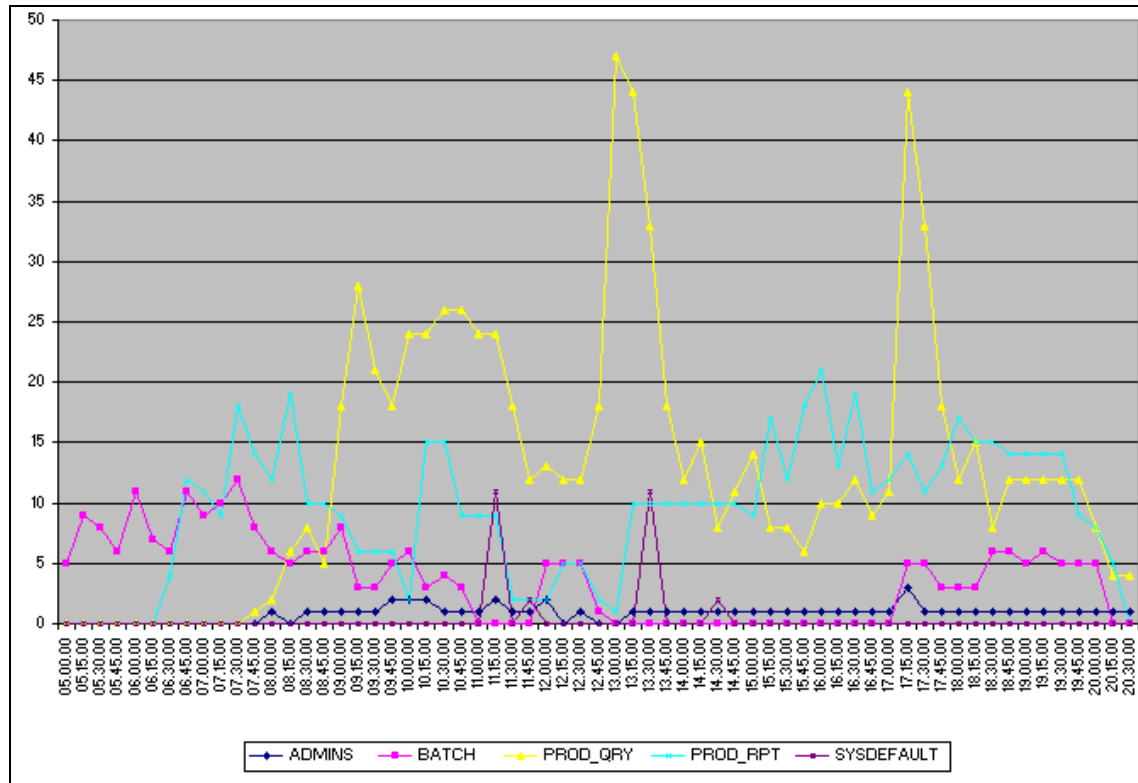


Figure 4-6 Active connections

Checking concurrent connections

We now turn our attention to concurrent connections to get a sense of the throughput of our work. This is shown in Figure 4-7 on page 78. Here we see the production reports and production queries are high during the mornings. The number of production reports and the CPU resources they consume may need to be controlled in order to protect the production query SLA. Again the SYSTEMDEFAULTSUBCLASS has connections most of the day.

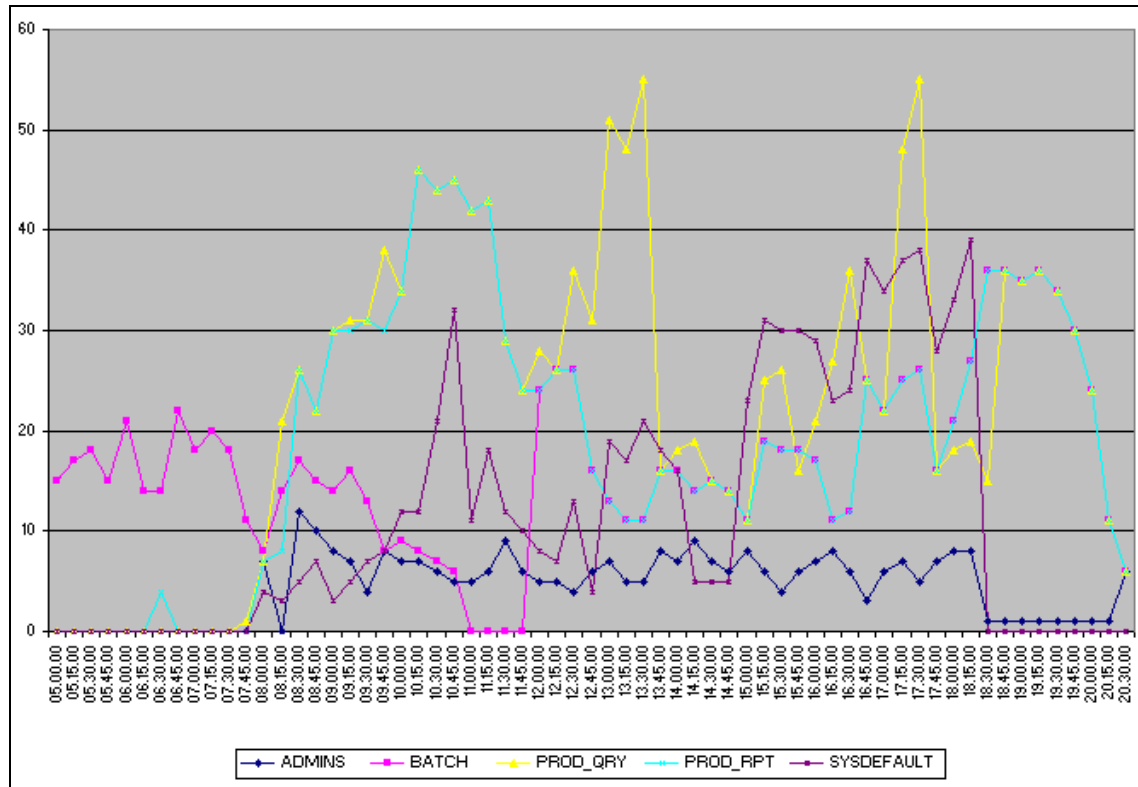


Figure 4-7 Concurrent connections

Using the following graph, we get an overall view of the number of connections hitting our system as shown in Figure 4-8 on page 79. We can see the high water mark for connections and that it occurs twice a day during prime shift.

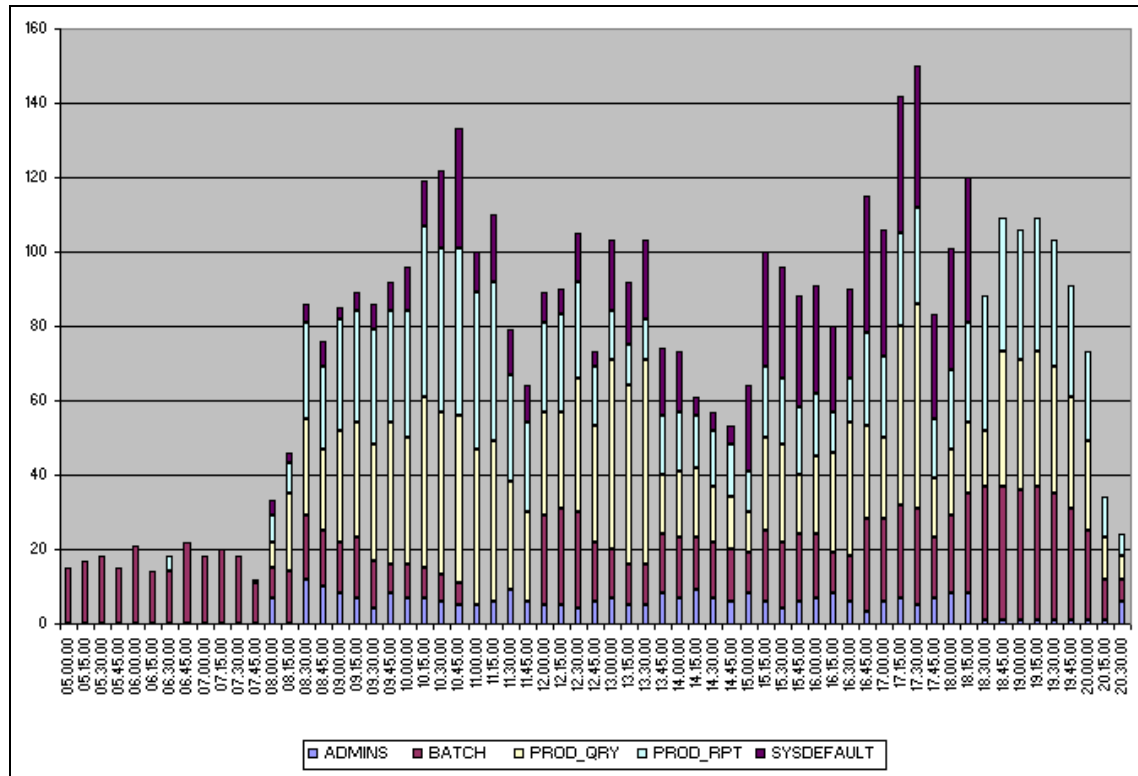


Figure 4-8 Connection high water mark

4.5 Summary

Now that we completed our cycle for the first time, what did we learn? The analysis of the data will be the input for the next cycle using our WLM methodology:

Identify

There are several areas we now want to identify for analysis.

- ▶ What is running in the SYSTEMDEFAULTSUBCLASS. We have already identified all existing categories. So this work is probably being performed by our report development group as it is the only remaining group allowed access to your example database.
- ▶ Limit the resources for the rogue queries based on execution time

- ▶ The BATCH group runs two types of workloads, LOAD and our ETL tool, *etl.exe* which we want to identify how much resources is being used for each of these categories.

Manage

In order to accomplish our new goals that we have identified we can alter our setup to the following Example 4-9.

Example 4-9 Altered basic setup

```
ALTER WORKLOAD w1_batch DISABLE;

ALTER SERVICE CLASS BATCH UNDER highlvl DISABLE;

DROP WORKLOAD w1_batch ;

DROP SERVICE CLASS batch UNDER highlvl;

CREATE SERVICE CLASS batch_load UNDER highlvl COLLECT AGGREGATE REQUEST DATA
BASE DISABLE;
CREATE SERVICE CLASS batch_etl UNDER highlvl COLLECT AGGREGATE REQUEST DATA
BASE DISABLE;
CREATE SERVICE CLASS dev_rpt UNDER highlvl COLLECT AGGREGATE ACTIVITY DATA
EXTENDED DISABLE;

CREATE WORKLOAD w1_batch_etl APPLNAME ('etl.exe') DISABLE SERVICE CLASS
batch_etl UNDER highlvl POSITION AT 2;
CREATE WORKLOAD w1_batch_load CURRENT CLIENT_USERID ('BATCH') DISABLE SERVICE
CLASS batch_load UNDER highlvl POSIION AT 3;
ALTER WORKLOAD w1_prod POSITION AT 4;
ALTER WORKLOAD w1_admin POSITION AT 5;
CREATE WORKLOAD w1_dev_rpt SESSION_USER GROUP ('DEVGRP') DISABLE SERVICE CLASS
HIGHLVL POSITION AT 6;

GRANT USAGE ON WORKLOAD w1_batch_etl TO PUBLIC;
GRANT USAGE ON WORKLOAD w1_batch_load TO PUBLIC;
GRANT USAGE ON WORKLOAD w1_dev_rpt TO PUBLIC;

ALTER SERVICE CLASS batch_etl UNDER highlvl ENABLE;
ALTER SERVICE CLASS batch_load UNDER highlvl ENABLE;

ALTER WORKLOAD w1_batch_etl ENABLE;
ALTER WORKLOAD w1_batch_load ENABLE;
ALTER WORKLOAD w1_dev_rpt ENABLE;

COMMIT;
--
-- SETUP THRESHOLD AND MONITORING
```



```
--  
  
CREATE THRESHOLD rouge_dev  
  FOR SERVICE CLASS dev_rpt UNDER highlvl ACTIVITIES  
  ENFORCEMENT DATABASE  
  WHEN ACTIVITYTOTALTIME > 17 MINUTES  
  COLLECT ACTIVITY DATA WITH DETAILS AND VALUES  
  STOP EXECUTION ;  
  
--  
-- Create Event Monitor  
--  
CREATE EVENT MONITOR VIOLATIONS FOR THRESHOLD VIOLATIONS WRITE TO TABLE  
  CONTROL (TABLE CNTL_VIOLATION IN MAINT),  
  THRESHOLDVIOLATIONS (TABLE THRESHOLD_VIOLATIONS IN MAINT) AUTOSTART;  
  
SET EVENT MONITOR VIOLATIONS STATE 1;
```

Monitor

We will continue using the same high level reporting and repeating our analysis and the WLM methodology cycle.



Monitoring

This chapter describes the methodology for monitoring the DB2 Workload Management (WLM) information. We discuss the monitoring tools in two categories: real-time monitoring and historical monitoring. Additionally, using a simple WLM configuration and those monitoring tools, we provide the monitoring examples and results for workload profiling and capturing.

In this chapter, the following topics are discussed:

- ▶ Real-time monitoring
 - Workload management table functions
 - Workload management stored procedures
 - db2pd command for workload management
- ▶ Historical monitoring
 - Activities event monitor
 - Threshold violations event monitor
 - Statistics event monitor
- ▶ Workload profiling and capturing

5.1 Real-time monitoring

Real-time monitoring is useful for you to see what happens on your system. This section describes how to use workload management table functions, stored procedures, and the **db2pd** command with examples. These monitoring tools enable you to access information for the new Workload Manager (WLM) objects.

5.1.1 Workload management table functions

DB2 9.5 provides new table functions for WLM to collect and report point-in-time workload information. These table functions do not use the existing system monitor or snapshot mechanisms, they access directly in-memory information, therefore, performance impact is minimum.

These table functions return monitor information, such as table data, by issuing an SQL query. Depending on what you want to monitor, you can customize the SQL easily.

All table functions can return information for either a single database partition or for all database partitions in a partitioned database environment. Those table functions have the `dbpartitionnum` input parameter. The parameter indicates -1 for the current database partition, or -2 for all database partitions.

DB2 9.5 provides two types of table functions for WLM: one for obtaining operational information and the other for obtaining statistics.

Table functions to obtain operational information

This set of table functions return the current workload condition and can be used to obtain the operational information:

► **WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES**

Table function parameters:

- `service_superclass_name`
- `service_subclass_name`
- `dbpartitionnum`

You can use this table function to get a list of workload occurrences, across partitions, assigned to a service class. For each occurrence, there is information about the current state, the connection attributes used to assign the workload to the service class, activity statistics indicating the activity volume, and the success rates. Example 5-1 shows how to use this table function to find out who is on the system.

Example 5-1 Who is on the system

```
>db2 "SELECT SUBSTR(service_superclass_name,1,19) AS superclass_name,
SUBSTR(service_subclass_name,1,19) AS subclass_name,
SUBSTR(workload_name,1,22) AS workload_name, application_handle,
workload_occurrence_state FROM
TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('','",-1))"
```

SUPERCLASS_NAME	SUBCLASS_NAME	WORKLOAD_NAME
APPLICATION_HANDLE	WORKLOAD_OCCURRENCE_STATE	
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	SYSDEFAULTUSERWORKLOAD
466	UOWEXEC	
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	SYSDEFAULTUSERWORKLOAD
513	UOWWAIT	
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	SYSDEFAULTUSERWORKLOAD
517	UOWEXEC	

3 record(s) selected.

The WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function does not report the applications which have the APPL_STATUS value CONNECTED. Because the applications haven't been assigned to a workload yet. You can use the SNAPAPPL_INFO administrative view to see all applications connected to DB2. See Example 5-2.

Example 5-2 SNAPAPPL_INFO administrative view

```
>db2 "SELECT agent_id,appl_status,substr(appl_name,1,10)
AS appl_name FROM sysibmadm.snapappl_info"
```

AGENT_ID	APPL_STATUS	APPL_NAME
13	CONNECTED	db2evm1_TO
12	CONNECTED	db2evm1_TO
11	CONNECTED	db2evmg_DB
466	UOWEXEC	db2bp.exe
10	CONNECTED	db2w1md
513	UOWWAIT	db2bp.exe
9	CONNECTED	db2taskd
8	CONNECTED	db2stmm
14	CONNECTED	db2evm1_TO
27	CONNECTED	db2bp.exe
517	UOWEXEC	db2bp.exe

11 record(s) selected.

► **WLM_GET_SERVICE_CLASS_AGENTS**

Table function parameters:

- service_superclass_name
- service_subclass_name
- application_handle
- dbpartitionnum

You can use this table function to obtain a list of agents working in the database. You can list all the agents running in a specific service class or all the agents working on the behalf of a particular application. You can also use this table function to determine the state of the coordinator agent and subagents for applications and determine which requests each agent in the system is working on.

The WLM_GET_SERVICE_CLASS_AGENTS table function reports the applications which have the APPL_STATUS value UOWEXEC.

Example 5-3 shows the event object (EVENT_OBJECT) and event state (EVENT_STATE) for the applications.

Example 5-3 Using WLM_GET_SERVICE_CLASS_AGENTS

```
>db2 "SELECT application_handle, uow_id, activity_id, event_object,
event_state FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('','cast(null as
bigint), -1)) WHERE service_superclass_name <> 'sysdefaultsystemclass'"
```

APPLICATION_HANDLE	UOW_ID	ACTIVITY_ID	EVENT_OBJECT	EVENT_STATE
466	68	1	ROUTINE	EXECUTING
517	3	1	REQUEST	EXECUTING

2 record(s) selected.

► **WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES**

Table function parameters:

- application_handle
- dbpartitionnum

You can use this table function to obtain a list of current activities that are associated with a workload occurrence. For each activity, the available information includes the current state of the activity (for example, executing or queued), the type of activity (for example, LOAD, READ, DDL), and the time at which the activity started. You can also identify who is running a long-running activity.

Example 5-4 shows two applications are reading DML.

Example 5-4 What is the application doing

```
>db2 "SELECT application_handle, local_start_time, activity_state,
activity_type FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(cast(null
as bigint), -1))"
```

APPLICATION_HANDLE	LOCAL_START_TIME	ACTIVITY_STATE	ACTIVITY_TYPE
466	2007-08-17-10.18.27.893233	EXECUTING	READ_DML
517	2007-08-17-10.17.08.991668	EXECUTING	READ_DML

2 record(s) selected.

► **WLM_GET_ACTIVITY_DETAILS**

Table function parameters:

- application_handle
- uow_id
- activity_id
- dbpartitionnum

You can use this table function to obtain the details about an individual activity of an application. For example, for SQL activities, the available information includes the statement text, package data, cost estimates, lock timeout value, and isolation level. Note that you need to turn on the statement monitor switch or the timestamp switch to collect some elements (for example, CPU times and rows returned or modified). The value -1 means that either the statement monitor switch or the timestamp switch is not activated.

Example 5-5 shows the SQL activities (isolation level, lock timeout, cost estimates) of the application handle 18.

Example 5-5 Capturing the individual activity

```
>db2 "SELECT SUBSTR(NAME, 1, 25) AS NAME, SUBSTR(VALUE, 1, 20) AS VALUE
FROM TABLE(WLM_GET_ACTIVITY_DETAILS(18,16,1,-2))
WHERE NAME IN ('COORD_PARTITION_NUM', 'APPLICATION_HANDLE',
'EFFECTIVE_ISOLATION', 'EFFECTIVE_LOCK_TIMEOUT', 'QUERY_COST_ESTIMATE')"
```

NAME	VALUE
APPLICATION_HANDLE	18
COORD_PARTITION_NUM	0
EFFECTIVE_ISOLATION	3
EFFECTIVE_LOCK_TIMEOUT	120
QUERY_COST_ESTIMATE	8

5 record(s) selected.

Identify lock wait

If you want to get the SQL statement using snapshot monitor, you need to turn on the statement monitor switch. WLM table functions provides you a way to obtain the running SQL statement without turning the monitoring switch on since it does not depend on the monitor switches.

Here we demonstrate how to use the table functions to identify the lock-wait query:

1. Identify the application handle using the SNAPAPPL_INFO administrative view as shown in Example 5-6. The application handle 43 is lock-waited.

Example 5-6 Identifying which application is lock-waited

```
>db2 "select agent_id,appl_status,substr(appl_name,1,10) AS appl_name from
sysibmadm.snapappl_info"
```

AGENT_ID	APPL_STATUS	APPL_NAME
39	CONNECTED	db2stmm
38	UOWWAIT	db2bp.exe
43	LOCKWAIT	java.exe
42	CONNECTED	db2evmg_DB
41	CONNECTED	db2w1md
47	UOWEXEC	db2bp.exe
40	CONNECTED	db2taskd

7 record(s) selected.

2. Identify the service superclass name and the service subclass name of the locked application using WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function.

Example 5-7 shows the application handle 43 is associated with the SYSDEFAULTSUBCLASS under the SYSDEFAULTUSERCLASS and is in UOWWAIT state.

Example 5-7 identifying the service superclass name and the service subclass name

```
>db2 "SELECT substr(service_superclass_name,1,19) as superclass_name,
substr(service_subclass_name,1,19) as subclass_name
, application_handle, workload_occurrence_state FROM
TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('','",-1))"
```

SUPERCLASS_NAME	SUBCLASS_NAME	APPLICATION_HANDLE
WORKLOAD_OCCURRENCE_STATE		


```

SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS          38
UOWWAIT
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS          43
UOWWAIT
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS          47
UOWEXEC

```

3 record(s) selected.

3. Identify the UOW ID and the activity ID using the WLM_GET_SERVICE_CLASS_AGENTS table function. Example 5-8 shows UOW ID and activity ID for application 43 respectively.

Example 5-8 Identifying the uow id and activity id

```

>db2 "SELECT application_handle, uow_id, activity_id, event_object,
event_state FROM
TABLE(WLM_GET_SERVICE_CLASS_AGENTS('SYSDEFAULTUSERCLASS','SYSDEFAULTSUBCLAS
S',43, -1)) as agents"

```

APPLICATION_HANDLE	UOW_ID	ACTIVITY_ID	EVENT_OBJECT	EVENT_STATE
43	3	1	LOCK	IDLE

1 record(s) selected.

4. Identify the SQL statement using the WLM_GET_ACTIVITY_DETAILS table function. Example 5-9 shows the query in lock-waited in STMT_TEXT. The STMT_TEXT field only contains the first 1024 characters of the statement text.

Example 5-9 Identifying the SQL statement

```

>db2 "SELECT substr(name, 1, 20) as name, substr(value, 1, 50) as value
FROM TABLE(WLM_GET_ACTIVITY_DETAILS(43,3,1,-1)) as actdetail WHERE NAME IN
('coord_partition_num', 'application_handle', 'local_start_time',
'application_handle', 'activity_id', 'uow_id', 'activity_state',
'activity_type', 'entry_time', 'local_start_time', 'stmt_text',
'rows_fetched', 'query_cost_estimate')"

```

NAME	VALUE
APPLICATION_HANDLE	43
COORD_PARTITION_NUM	0
UOW_ID	3
ACTIVITY_ID	1
ACTIVITY_STATE	EXECUTING
ACTIVITY_TYPE	READ_DML
ENTRY_TIME	2007-08-16-12.04.04.744903
LOCAL_START_TIME	2007-08-16-12.04.04.744927

```

STMT_TEXT          SELECT NAME FROM STAFF WHERE ID = ?
QUERY_COST_ESTIMATE 8
ROWS_FETCHED       -1

```

11 record(s) selected.

Table functions to obtain statistics

This set of table functions return the detailed workload information since the last time that the statistics were reset. These table functions report only a subset of the statistics. To view the full set of statistics, you must collect the statistics information.

Table functions for obtaining statistics are:

► **WLM_GET_SERVICE_SUPERCLASS_STATS**

Table function parameters:

- service_superclass_name
- dbpartitionnum

You can use this table function to show summary statistics across partitions at the service superclass level. For example, high water marks for concurrent connections which is useful when determining peak workload activity.

Example 5-10 shows the concurrent connections for each service superclass.

Example 5-10 Using WLM_GET_SERVICE_SUPERCLASS_STATS

```
>db2 "SELECT * FROM TABLE(WLM_GET_SERVICE_SUPERCLASS_STATS(' ', -1))"
```

SERVICE_SUPERCLASS_NAME	DBPARTITIONNUM	LAST_RESET
CONCURRENT_CONNECTION_TOP		
-----	-----	-----
SYSDEFAULTSYSTEMCLASS	0	2007-08-17-08.25.53.703291
7		
SYSDEFAULTMAINTENANCECLASS	0	2007-08-17-08.25.53.703367
2		
SYSDEFAULTUSERCLASS	0	2007-08-17-08.25.53.703419
15		

3 record(s) selected.

► **WLM_GET_SERVICE_SUBCLASS_STATS**

Table function parameters:

- service_superclass_name

- service_subclass_name
- dbpartitionnum

You can use this table function to show summary statistics across partitions at the service subclass level (all activities run in service subclasses). Statistics includes numbers of activities and average execution times. The average execution times is useful when looking at general system health and the distribution of activities across service classes and partitions.

Lifetime information such as average execution times is only returned for those service classes that are defined with COLLECT AGGREGATE ACTIVITY DATA. While the SYSDEFAULTSUBCLASS under the SYSDEFAULTUSERCLASS is not defined with COLLECT AGGREGATE ACTIVITY DATA by default, the value of REQUEST_EXEC_TIME_AVG is NULL.

Example 5-11 shows the number of requests (NUM_REQUESTS_ACTIVE) that are executing in the service subclass.

Example 5-11 Using WLM_GET_SERVICE_SUBCLASS_STATS

```
>db2 "SELECT substr(service_superclass_name,1,19) as superclass_name,
substr(service_subclass_name,1,18) as subclass_name, num_requests_active,
request_exec_time_avg FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('','', -1))
ORDER BY superclass_name, subclass_name"
```

SUPERCLASS_NAME	SUBCLASS_NAME	NUM_REQUESTS_ACTIVE
REQUEST_EXEC_TIME_AVG		
-----	-----	-----
-----	-----	-----
SYSDEFAULTMAINTENAN	SYSDEFAULTSUBCLASS	0
-		
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	5
-		
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	15
-		

3 record(s) selected.

► **WLM_GET_WORK_ACTION_SET_STATS**

Table function parameters:

- work_action_set_name
- dbpartitionnum

You can use this table function to show summary statistics across partitions at the work action set level, namely, the number of activities assigned to a work action set and the related work class name. This is useful for understanding the effectiveness of a work action set and understanding the types of activities executing on the system.

In Example 5-11, the value 3 means that three activities have been assigned to the work class given by WORK_CLASS_NAME since the last reset.

Example 5-12 Using WLM_GET_WORK_ACTION_SET_STATS

```
> db2 "SELECT substr(work_action_set_name,1,18) as work_action_set_name,
substr(work_class_name,1,15) as work_class_name,
substr(char(act_total),1,14) as act_total, last_reset FROM
TABLE(WLM_GET_WORK_ACTION_SET_STATS (cast(null as varchar(128)), -1)) as
wasstats order by work_action_set_name, work_class_name"
```

WORK_ACTION_SET_NAME	WORK_CLASS_NAME	ACT_TOTAL	LAST_RESET
SPECIFIC_ACTIONSET	*	0	2007-08-24-18.22.02.038311
SPECIFIC_ACTIONSET	SPECIFIC_QUERY	3	2007-08-24-18.22.02.038311

2 record(s) selected.

► **WLM_GET_WORKLOAD_STATS**

Table function parameters:

- workload_name
- dbpartitionnum

You can use this table function to show summary statistics across partitions at the workload level. This includes high water marks for concurrent workload occurrences and numbers of completed activities. This is useful when monitoring general system health or drilling down to identify problem areas.

Example 5-13 shows the highest number of concurrent occurrences and the highest number of concurrent activities for each workload. The CONCURRENT_WLO_ACT_TOP field is updated by each workload occurrence at the end of its unit of work.

Example 5-13 Using WLM_GET_WORKLOAD_STATS

```
>db2 "SELECT substr(workload_name,1,22) as workload_name,
concurrent_wlo_top, concurrent_wlo_act_top FROM
TABLE(WLM_GET_WORKLOAD_STATS(CAST(null as varchar(128)), -1)) ORDER BY
workload_name"
```

WORKLOAD_NAME	CONCURRENT_WLO_TOP	CONCURRENT_WLO_ACT_TOP
SYSDEFAULTADMWORKLOAD	0	0
SYSDEFAULTUSERWORKLOAD	16	2

2 record(s) selected.

► **WLM_GET_QUEUE_STATS**

Table function parameters:

- threshold_predicate
- threshold_domain
- threshold_name
- threshold_id

You can use this table function to show summary statistics across partitions for the WLM queues used for their corresponding thresholds. Statistics includes the number of queued activities (current and total) and total time spent in a queue. This is useful when querying current queued activity or validating if a threshold is correctly defined. Excessive queuing might indicate that a threshold is too restrictive, and very little queuing might indicate that a threshold is not restrictive enough or is not needed.

Example 5-14 shows that the threshold QUEUE_THRESH is a concurrent database coordinator activities (CONCDBC) threshold. It defines an upper bound of the number of concurrent coordinator activities. The domain of the threshold is SB (service subclass). Total 17 activities were assigned to this queue since the last reset. 806299 milliseconds spent in the queue for 17 activities.

Example 5-14 Using WLM_GET_QUEUE_STATS

```
>db2 "SELECT substr(threshold_name, 1, 15) threshname, threshold_predicate,
threshold_domain, dbpartitionnum part, queue_size_top, queue_size_current,
queue_time_total, queue_assignments_total queue_assign FROM
table(WLM_GET_QUEUE_STATS(' ', ' ', ' ', -1))"
```

THRESHNAME	THRESHOLD_PREDICATE	THRESHOLD_DOMAIN	PART
QUEUE_SIZE_TOP	QUEUE_SIZE_CURRENT	QUEUE_TIME_TOTAL	QUEUE_ASSIGN
-----	-----	-----	-----
QUEUE_THRESH	CONCDBC	SB	0
10	10	806299	17
1 record(s) selected.			

Workload management and snapshot monitor integration

Snapshot™ monitor is also useful for you to capture the condition of database system. You can use both snapshot monitor table functions and SQL administrative views with workload management table functions at the same time. There are fields you can use when joining table functions between workload management and snapshot monitor.

Table 5-1 lists the fields shared between the workload management table functions and the snapshot monitor table functions.

Table 5-1 WLM and snapshot monitor table functions field mapping

WLM table function field	Snapshot monitor table function field
agent_tid	agent_pid
application_handle	agent_id agent_id_holding_lock
session_auth_id	session_auth_id
dbpartitionnum	node_number
utility_id	utility_id
workload_id	workload_id

Note: Snapshot monitor table functions may be modified in the future DB2 release. While the set of snapshot administrative views will remain the same with new columns added to the view. It is good for application maintenance to use snapshot monitor SQL administrative views.

Example 5-15 shows how to identify applications CPU consumption by joining the WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function and the SNAPAPPL administrative view. In this case, the AGENT_ID 282 consumes the most CPU resources.

Example 5-15 Identifying applications consuming CPU resource

```
db2 "SELECT substr(wlm.service_superclass_name,1,25) as superclass_name,
substr(wlm.service_subclass_name,1,25) as subclass_name,
substr(wlm.workload_name,1,25) as workload_name, snap.agent_id,
sum(snap.agent_usr_cpu_time_s + snap.agent_sys_cpu_time_s) as cpu_time FROM
sysibmadm.snapappl snap,
TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('','",-1)) wlm WHERE
snap.agent_id=wlm.application_handle GROUP BY snap.agent_id,
wlm.service_superclass_name, wlm.service_subclass_name, wlm.workload_name ORDER
BY cpu_time DESC FETCH FIRST 5 ROWS ONLY"
```

SUPERCLASS_NAME	SUBCLASS_NAME	WORKLOAD_NAME
AGENT_ID	CPU_TIME	
-----	-----	-----
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	SYSDEFAULTUSERWORKLOAD
282	26	
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	SYSDEFAULTUSERWORKLOAD
7	13	

SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	SYSDEFAULTUSERWORKLOAD
17	2	
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	SYSDEFAULTUSERWORKLOAD
16	0	
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	SYSDEFAULTUSERWORKLOAD
212	0	

5 record(s) selected.

5.1.2 Workload management stored procedures

DB2 9.5 provides new stored procedures for WLM to manage activities. Each stored procedure offers a specific functionality.

WLM stored procedures are:

► WLM_CANCEL_ACTIVITY

Syntax:

```
WLM_CANCEL_ACTIVITY (application_handle, uow_id, activity_id)
```

You can use this stored procedure to cancel a running or queued activity. You identify the activity by its application handle, unit of work identifier, and activity identifier. You can cancel any type of activity but not force the connection. The application with the canceled activity receives the error SQL4725N.

In Example 5-16, from window A, we first tried to cancel application 637 with UOW ID 3. This is not an exiting activity and DB2 returns SQL4702N with SQLSTATE 5U035. We then successfully cancelled application 637, UOW ID 1. In WindowB, we see the application is canceled but the connection stays.

Example 5-16 Canceling an activity

WindowA:

```
>db2 CALL WLM_CANCEL_ACTIVITY(637,3,1)
```

```
SQL4702N The activity identified by application handle "637 [0-637]", unit of work ID "3", and activity ID "1" does not exist. SQLSTATE=5U035
```

```
>db2 CALL WLM_CANCEL_ACTIVITY(637,1,1)
```

```
Return Status = 0
```

WindowB:

```
>db2 "SELECT * FROM employee FOR UPDATE"
```

EMPNO	FIRSTNAME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE	JOB
EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM		

```
-----
-----
-----
SQL4725N The activity has been cancelled.  SQLSTATE=57014
```

```
>db2 get connection state
```

Database Connection State

```
Connection state      = Connectable and Connected
Connection mode       = SHARE
Local database alias  = SAMPLE
Database name         = SAMPLE
Hostname              =
Service name          =
```

► WLM_CAPTURE_ACTIVITY_IN_PROGRESS

Syntax:

```
WLM_CAPTURE_ACTIVITY_IN_PROGRESS(application_handle, uow_id, activity_id)
```

You can use this stored procedure to send information about an individual activity that is currently executing to the active activities event monitor. This stored procedure sends the information immediately, rather than waiting until the activity completes. Before you call this stored procedure, you need to activate an activity event monitor. If there is no active activities event monitor, an SQL1633W with SQLSTATE 01H53 is returned.

Example 5-17 shows that SQL1633 is returned on the first stored procedure call due to the event monitor is not activated. Once the event monitor is on, the stored procedure run successfully.

Example 5-17 Capturing an activity

```
>db2 CALL WLM_CAPTURE_ACTIVITY_IN_PROGRESS(660,2,1)
```

```
Return Status = 0
```

```
SQL1633W The activity identified by application handle "660 [0-660]", unit of work ID "2", and activity ID "1" could not be captured because there is no active activity event monitor.  SQLSTATE=01H53
```

```
>db2 set event monitor act state 1
```

```
DB20000I The SQL command completed successfully.
```

```
>db2 CALL WLM_CAPTURE_ACTIVITY_IN_PROGRESS(660,2,1)
```

```
Return Status = 0
```

► **WLM_COLLECT_STATS**

Syntax:

```
WLM_COLLECT_STATS()
```

You can use this stored procedure to collect and reset statistics for workload management objects. All statistics tracked for service classes, workloads, threshold queues, and work action sets are sent to the active statistics event monitor (if one exists) and then reset. If there is no active statistics event monitor, the statistics are only reset, but not collected.

Example 5-18 shows that the data of last_reset field is changed after calling the WLM_COLLECT_STATS stored procedure.

Example 5-18 calling the WLM_COLLECT_STATS stored procedure

```
>db2 "SELECT last_reset FROM
TABLE(WLM_GET_SERVICE_SUPERCLASS_STATS('SYSDEFAULTUSERCLASS',-1))"
```

```
LAST_RESET
-----
2007-08-15-14.56.08.129076
```

```
1 record(s) selected.
```

```
>db2 "CALL WLM_COLLECT_STATS()"
```

```
Return Status = 0
```

```
>db2 "SELECT last_reset FROM
TABLE(WLM_GET_SERVICE_SUPERCLASS_STATS('SYSDEFAULTUSERCLASS',-1))"
```

```
LAST_RESET
-----
2007-08-15-14.57.31.639612
```

```
1 record(s) selected.
```

5.1.3 db2pd command for workload management

In DB2 9.5, db2pd command is enhanced for workload management to return operational information from the DB2 database system memory sets. Compared to table functions, db2pd command is easier to use since you don't need to prepare the SQL statements with table functions.

Db2pd command with various options for workload management are:

- ▶ `db2pd -workloads [none/workloadID]`
Returns a list of workload definitions in memory at the time the command is run.
- ▶ `db2pd -serviceclasses [none/serviceclassID]`
Returns information about the service classes for a database. `serviceclassID` is an optional parameter to retrieve information for one specific service class. If `serviceclassID` is not specified, information for all service classes is retrieved.
- ▶ `db2pd -workactionsets [none/workactionsetID]`
Returns information about all enabled work action sets, as well as all the enabled work actions in the enabled work action sets.
- ▶ `db2pd -workclasssets [none/workclasssetID]`
Returns information about all work class sets that have been referenced by an enabled work action set, as well as all work classes in those work class sets.
- ▶ `db2pd -threshold [none/thresholdID]`
Returns information about thresholds. `thresholdID` is optional. Specifying a threshold ID returns information about a specific threshold. If `thresholdID` is not specified, information for all thresholds is retrieved.

Example 5-19 shows using `db2pd` to retrieve information for service class ID 3 in `SAPMPLE` database. You can see the application handle list associated with the service class.

Example 5-19 using db2pd command

```
>db2pd -db sample -serviceclasses 3
```

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 02:30:04
```

```
Service Classes:
```

```
Service Class Name      = SYSDEFAULTUSERCLASS
Service Class ID       = 3
Service Class Type     = Service Superclass
Default Subclass ID   = 13
Service Class State    = Enabled
Agent Priority         = Default
Prefetch Priority      = Default
Outbound Correlator   = None
Work Action Set ID    = N/A
Collect Activity Opt  = None
```

```

Num Connections          = 3
Last Statistics Reset Time = 2007-08-17 08:25:53.000000
Num Coordinator Connections = 3
Coordinator Connections HWM = 4

```

Associated Workload Occurrences (WLO):

AppHandl	[nod-index]	WL ID	WLO ID	UOW ID	WLO State
466	[000-00466]	1	1	116	UOWWAIT
513	[000-00513]	1	147	11	UOWWAIT
660	[000-00660]	1	295	3	UOWEXEC

Note: You also can get objectID from system catalog views. Here are the examples:

```

SELECT workloadid,workloadname FROM syscat.workloads
SELECT serviceclassid,serviceclassname FROM syscat.serviceclasses
SELECT actionsetid,actionsetname FROM syscat.workactionsets
SELECT workclassid,workclassname FROM syscat.workclasses
SELECT thresholdid,thresholdname FROM syscat.thresholds

```

5.2 Historical monitoring

Historical monitoring is useful for seeing what have happened on your system. DB2 provides three event monitor types (ACTIVITIES, THRESHOLD VIOLATIONS and STATISTICS) for WLM to store activity information or aggregate information. This section describes how to collect and analyze those information.

Typically, event monitors write data to either tables or files. Compared to real-time monitoring, event monitors consume disk space to store information in tables or files space. You need to prune these tables or files periodically because they are not automatically pruned. The event monitor file output requires formatting using db2evmon in order to be readable. While the event monitor table can be queried and manipulated with SQL, the event monitor table is easier for historical analysis.

You can use the wlmvmon.ddl script in the sqllib/misc directory to create and enable three event monitors DB2ACTIVITIES, DB2STATISTICS, and DB2THRESHOLDVIOLATIONS. These event monitors write data to tables. If you execute the script without modification, tables are stored in the USERSPACE1. If necessary, modify the script to change the table space or other parameters.

Table 5-2 shows the event monitor type for WLM and event group. If you create a activity event monitor write data to tables, the database creates 4 target tables to store records for each of the event groups.

Table 5-2 *Type of event monitor and event monitor group value*

Type of Event Monitor	evm-group Value
Activities	CONTROL ACTIVITY ACTIVITYSTMT ACTIVITYVALS
Threshold Violations	CONTROL THRESHOLDVIOLATIONS
Statistics	CONTROL SCSTATS WCSTATS WLSTATS HISTOGRAMBIN QSTATS

Note: Only one of the event monitor ACTIVITIES, STATISTICS, or THRESHOLD VIOLATIONS can be activate at any one time. If an event monitor of the same type is already active, SQL1631N with SQLSTATE 5U024 is returned.

5.2.1 Activities event monitor

This section introduces how to use the activities event monitor. The activities event monitor has the following features:

- ▶ Which WLM objects can I collect activity information for?
 - You can collect information about individual activities for service subclasses, workloads, work classes (through work actions), and threshold violations.
- ▶ When is the information useful?
 - As input to tools such as Design advisor(db2advis) or explain utility (to obtain access plans)
 - Debug individual activities.
- ▶ When is the information collected?
 - The information is collected when the activity completes, regardless of whether the activity completes successfully. When an activity completes,

information about the activity is sent to the active ACTIVITIES event monitor, if one exists.

- ▶ How do I collect the information?

You can collect information about an activity by specifying COLLECT ACTIVITY DATA for the service subclass, workload, or work action to which such an activity belongs or a threshold that might be violated by such an activity.

Collecting WLM object activity information

The following are steps to enable collection of activities for a given workload management object:

1. Create an event monitor:

Use the CREATE EVENT MONITOR statement to create an ACTIVITIES type event monitor.

Example 5-20 shows how to create an activities event monitor that writes data to a file.

Example 5-20 Creating an activities event monitor to a file

```
$ db2 "create event monitor wlm_act for activities write to file
'/evmon/wlm/activities' autostart"
DB20000I The SQL command completed successfully.
$ db2 commit
DB20000I The SQL command completed successfully.
```

Example 5-21 shows to how to create an activities event monitor that writes data to tables. Four tables are created.

Example 5-21 Creating an activities event monitor tables

```
$ db2 "create event monitor wlm_act for activities write to table activity
(table activity_db2activities in userspace1), activitystmt (table
activitystmt_db2activities in userspace1), activityvals (table
activityvals_db2activities in userspace1), control (table
control_db2activities in userspace1) autostart"
DB20000I The SQL command completed successfully.
$ db2 commit
DB20000I The SQL command completed successfully.
```

If you want to know the column name or the type name for those event monitor tables, you can use the DESCRIBE TABLE command, for example:

```
db2 describe table activity_db2activities
```

2. Activate the event monitor:

Use the SET EVENT MONITOR STATE statement to activate the event monitor.

Example 5-22 shows how to set the event monitor to active and check its status.

Example 5-22 Setting the event monitor to active

```
$ db2 "SELECT substr(evmonname,1,20) as evmonname, CASE WHEN
event_mon_state(evmonname) = 0 THEN 'Inactive' WHEN
event_mon_state(evmonname) = 1 THEN 'Active' END FROM syscat.eventmonitors"
```

```
EVMONNAME          2
-----
DB2DETAILDEADLOCK  Active
WLM_ACT            Inactive
```

2 record(s) selected.

```
$ db2 set event monitor wlm_act state 1
DB20000I The SQL command completed successfully.
```

```
$ db2 commit
DB20000I The SQL command completed successfully.
```

```
$ db2 "SELECT substr(evmonname,1,20) as evmonname, CASE WHEN
event_mon_state(evmonname) = 0 THEN 'Inactive' WHEN
event_mon_state(evmonname) = 1 THEN 'Active' END FROM syscat.eventmonitors"
```

```
EVMONNAME          2
-----
DB2DETAILDEADLOCK  Active
WLM_ACT            Active
```

2 record(s) selected.

3. Identify the target objects:

Identify the objects for which you want to collect activities by using the ALTER SERVICE CLASS, ALTER WORK ACTION SET, ALTER THRESHOLD, or ALTER WORKLOAD statement and specify the COLLECT ACTIVITY DATA keywords.

Example 5-23 shows how to alter the default service class to specify the COLLECT ACTIVITY DATA keyword. The COLLECT ACTIVITY DATA clause is only valid for a service subclass. The value W means activity data without details should be collected by the applicable event monitor.

Example 5-23 Using the ALTER SERVICE CLASS

```
$ db2 "SELECT substr(serviceclassname,1,26) as
serviceclassname,substr(parentserviceclassname,1,28) as
superclassname,collectactdata FROM syscat.serviceclasses"
```

SERVICECLASSNAME	SUPERCLASSNAME	COLLECTACTDATA
SYSDEFAULTSUBCLASS	SYSDEFAULTSYSTEMCLASS	N
SYSDEFAULTSUBCLASS	SYSDEFAULTMAINTENANCECLASS	N
SYSDEFAULTSUBCLASS	SYSDEFAULTUSERCLASS	N
SYSDEFAULTSYSTEMCLASS	-	N
SYSDEFAULTMAINTENANCECLASS	-	N
SYSDEFAULTUSERCLASS	-	N

6 record(s) selected.

```
$ db2 "alter service class SYSDEFAULTSUBCLASS under SYSDEFAULTUSERCLASS
collect activity data on all without details"
DB20000I The SQL command completed successfully.
```

```
$ db2 commit
DB20000I The SQL command completed successfully.
```

```
$ db2 "SELECT substr(serviceclassname,1,26) as
serviceclassname,substr(parentserviceclassname,1,28) as
superclassname,collectactdata FROM syscat.serviceclasses"
```

SERVICECLASSNAME	SUPERCLASSNAME	COLLECTACTDATA
SYSDEFAULTSUBCLASS	SYSDEFAULTSYSTEMCLASS	N
SYSDEFAULTSUBCLASS	SYSDEFAULTMAINTENANCECLASS	N
SYSDEFAULTSUBCLASS	SYSDEFAULTUSERCLASS	W
SYSDEFAULTSYSTEMCLASS	-	N
SYSDEFAULTMAINTENANCECLASS	-	N
SYSDEFAULTUSERCLASS	-	N

6 record(s) selected.

Three levels of workload capture

You can set the level of activities event monitor by specifying COLLECT ACTIVITY DATA keywords. For each WLM object, you can also activate at different levels and with different settings.

There are three levels for capturing workload activities:

- ▶ Default (COLLECT ACTIVITY DATA WITHOUT DETAILS option)

This level collects default information including WLM identification and basic time statistics.

▶ Detailed (COLLECT ACTIVITY DATA WITH DETAILES option)

This level collects detailed information includes statement text (static and dynamic SQL) and compilation environment.

▶ Detailed with input data values (COLLECT ACTIVITY DATA WITH DETAILES AND VALUES option)

This level collects default information, detailed information, as well as data values.

We show you the difference between these three levels by examples.

In Example 5-24, we create service class, subclasses, and workload to collect Java™ application activity information. Three creating workload statement are shown for collecting different level of details. Each time, only one of the statement is run. We also create an activity event monitor that writes data to a file.

Example 5-24 Commands to use COLLECT ACTIVITY DATA keywords

```
$ db2 "create service class actservice"
DB20000I The SQL command completed successfully.

$ db2 "create service class subactservice under actservice"
DB20000I The SQL command completed successfully.

$ db2 "create workload actworkload applname('java.exe') service class
subactservice under actservice collect activity data without details"
or
$ db2 "create workload actworkload applname('java.exe') service class
subactservice under actservice collect activity data with details"
or
$ db2 "create workload actworkload applname('java.exe') service class
subactservice under actservice collect activity data with details and values"
DB20000I The SQL command completed successfully.

$ db2 "create event monitor wlm_act for activities write to file
'/evmon/wlm/activities' autostart"

-----A java application is run.-----

$ db2evmon -path /evmon/wlm/activities > db2evmon.out
```

Example 5-25 shows the output of activities event monitor for workload APPLNAME with COLLECT ACTIVITY DATA WITHOUT DETAILES option.

Example 5-25 COLLECT ACTIVITY DATA WITHOUT DETAILES option output

```

6) Activity ...
   Activity ID                : 1
   Activity Secondary ID      : 0
   Appl Handle                : 113
   UOW ID                     : 1
   Service Superclass Name    : ACTSERVICE
   Service Subclass Name      : SUBACTSERVICE

   Activity Type              : READ_DML
   Parent Activity ID         : 0
   Parent UOW ID              : 0
   Coordinating Partition     : 0
   Workload ID                : 3
   Workload Occurrence ID     : 2
   Database Work Action Set ID : 0
   Database Work Class ID     : 0
   Service Class Work Action Set ID : 0
   Service Class Work Class ID : 0
   Time Created                : 2007-08-20 18:20:31.262589
   Time Started                : 2007-08-20 18:20:31.262602
   Time Completed              : 2007-08-20 18:20:31.262887
   Activity captured while in progress: FALSE

   Application ID              : *LOCAL.DB2_01.070821000549
   Application Name            : java.exe
   Session Auth ID            : db2inst1
   Client Userid               :
   Client Workstation Name     : DB2_COMP
   Client Applname             :
   Client Accounting String    :
   SQLCA:
   SQL0100W No row was found for FETCH, UPDATE or DELETE; or the result of a
   query is an empty table.  SQLSTATE=02000

   Query Cost Estimate        : 8
   Query Card Estimate        : 1
   Execution time              : 0.000284 seconds
   Rows Returned               : 1

```

Example 5-26 shows the output of activities event monitor or workload APPLNAME with COLLECT ACTIVITY DATA WITH DETAILES option. The activity statement section is added. From Statement text, we can identify which query is being run by Appl Handle 352.

Example 5-26 COLLECT ACTIVITY DATA WITH DETAILES option output

```

5) Activity ...

```

```

Activity ID                : 1
Activity Secondary ID     : 0
Appl Handle               : 352
UOW ID                    : 2
Service Superclass Name   : ACTSERVICE
Service Subclass Name     : SUBACTSERVICE

Activity Type              : READ_DML
Parent Activity ID        : 0
Parent UOW ID             : 0
Coordinating Partition    : 0
Workload ID               : 3
Workload Occurrence ID    : 1
Database Work Action Set ID : 0
Database Work Class ID    : 0
Service Class Work Action Set ID : 0
Service Class Work Class ID : 0
Time Created              : 2007-08-20 17:58:44.740640
Time Started              : 2007-08-20 17:58:44.740667
Time Completed            : 2007-08-20 17:58:44.741260
Activity captured while in progress: FALSE

```

```

Application ID             : *LOCAL.DB2_01.070821044851
Application Name           : java.exe
Session Auth ID           : db2inst1
Client Userid              :
Client Workstation Name   : DB2_COMP
Client Applname            :
Client Accounting String   :
SQLCA:

```

SQL0100W No row was found for FETCH, UPDATE or DELETE; or the result of a query is an empty table. SQLSTATE=02000

```

Query Cost Estimate       : 8
Query Card Estimate       : 1
Execution time            : 0.000592 seconds
Rows Returned             : 1

```

6) Activity Statement ...

```

Activity ID                : 1
Activity Secondary ID     : 0
Application ID             : *LOCAL.DB2_01.070821044851
UOW ID                    : 2

Lock timeout value        : -1
Query ID                  : 0
Package cache ID          : 103079215105
Package creator            : NULLID
Package name               : SYSSH200

```

```

Package version      :
Section No          : 1
Type                : Dynamic
Nesting level of stmt : 0
Source ID           : 0
Isolation level     : Cursor Stability
Statement text      : SELECT NAME FROM STAFF WHERE ID = ?

Stmt first use time : 2007-08-20 17:58:44.740640
Stmt last use time  : 2007-08-20 17:58:44.740640

```

Example 5-27 shows the output of activities event monitor or workload APPLNAME with COLLECT ACTIVITY DATA WITH THE DETAILES AND VALUES option. The activity data section is added. We can identify the value of parameter marker. Before DB2 9.5, even if you use the statement event monitor, you can not collect the value of parameter marker.

Example 5-27 COLLECT ACTIVITY DATA WITH DETAILES AND VALUES option output

```

5) Activity ...
  Activity ID                : 1
  Activity Secondary ID      : 0
  Appl Handle                 : 360
  UOW ID                     : 2
  Service Superclass Name    : ACTSERVICE
  Service Subclass Name      : SUBACTSERVICE

  Activity Type              : READ_DML
  Parent Activity ID         : 0
  Parent UOW ID              : 0
  Coordinating Partition    : 0
  Workload ID                : 3
  Workload Occurrence ID    : 1
  Database Work Action Set ID : 0
  Database Work Class ID     : 0
  Service Class Work Action Set ID : 0
  Service Class Work Class ID : 0
  Time Created               : 2007-08-20 18:13:28.207238
  Time Started               : 2007-08-20 18:13:28.207265
  Time Completed             : 2007-08-20 18:13:28.207860
  Activity captured while in progress: FALSE

  Application ID             : *LOCAL.DB2_01.070821045412
  Application Name           : java.exe
  Session Auth ID           : db2inst1
  Client Userid              :

```

Client Workstation Name : DB2_COMP
 Client Applname :
 Client Accounting String :

SQLCA:

SQL0100W No row was found for FETCH, UPDATE or DELETE; or the result of a query is an empty table. SQLSTATE=02000

Query Cost Estimate : 8
 Query Card Estimate : 1
 Execution time : 0.000595 seconds
 Rows Returned : 1

6) Activity Statement ...

Activity ID : 1
 Activity Secondary ID : 0
 Application ID : *LOCAL.DB2_01.070821045412
 UOW ID : 2

Lock timeout value : -1
 Query ID : 0
 Package cache ID : 103079215105
 Package creator : NULLID
 Package name : SYSSH200
 Package version :
 Section No : 1
 Type : Dynamic
 Nesting level of stmt : 0
 Source ID : 0
 Isolation level : Cursor Stability
 Statement text : SELECT NAME FROM STAFF WHERE ID = ?

 Stmt first use time : 2007-08-20 18:13:28.207238
 Stmt last use time : 2007-08-20 18:13:28.207238

7) Activity data values...

Activity ID : 1
 Activity Secondary ID : 0
 Application ID : *LOCAL.DB2_01.070821045412
 UOW ID : 2

 Value position : 1
 Value type : SMALLINT
 Value set by reopt : FALSE
 Value is NULL : FALSE
 Value data : 10

Importing activity information into the Design Advisor

The design advisor is a useful tool for tuning SQL statement performance. It recommends indexes for the SQL statements that user provides. In DB2 9.5, You can provide SQL statements by importing activity information from a workload or a service class into the Design Advisor.

Before importing activity information into the Design Advisor, the following prerequisites must be satisfied:

- ▶ An activity event monitor table is existed and activity information stored. You can not import information from activity event monitor files.
- ▶ Activities must have been collected using the COLLECT ACTIVITY DATA WITH DETAILS or COLLECT ACTIVITY DATA WITH DETAILS AND VALUES options.
- ▶ Explain tables exist
You can use the EXPLAIN.DDL script in the sqllib/misc directory to create the explain tables.

You can import activity information into the Design Advisor using db2advis command, for example:

```
db2advis -d sample -wlm db2activities workloadname actworkload
```

This example shows to import information about ACTWORKLOAD activities collected by DB2ACTIVITIES event monitor in the SAMPLE database. You can specify the workload or service class name and the start time and end time.

5.2.2 Threshold violations event monitor

This section introduces how to use the threshold violations event monitor. The following are the features of threshold violations event monitor:

- ▶ Which WLM objects can I collect threshold violation information for?
You can collect information for individual activities for threshold violations.
- ▶ When is the information useful?
The information is useful for identifying
 - The activity that violated the threshold, including the application handle and application ID.
 - Which threshold was violated
 - The time that the threshold was violated
 - The action that was taken (stop or continue)
- ▶ How do I collect the information?

- Threshold violations event monitor

If an threshold violations event monitor is created and activated, by default, information (for example, threshold id, time of violation threshold action and so on) about the activities that violate the threshold is collected. You don't need to specify COLLECT ACTIVITY DATA keywords for the threshold violations event monitor.

- Activity event monitor

You can optionally have activity information (for example, SQLCODE, SQLSTATE, execution time, and so on) written to an active activity event monitor if the threshold violation is caused by an activity. You need to specify COLLECT ACTIVITY DATA keywords for the activity event monitor.

- ▶ When is the information collected?

- Threshold violations event monitor

When a workload management threshold is violated, a threshold violation record is written to the active THRESHOLD VIOLATIONS event monitor, if one exists.

- Activity event monitor

The activity information is written when the activity completes, not when the threshold is violated.

Collecting threshold violations

Use the following steps to collect threshold violations:

1. Create an event monitor:

Use the CREATE EVENT MONITOR statement to create an THRESHOLD VIOLATIONS event monitor.

Example 5-28 shows how to create a threshold violations event monitor that writes data to a file.

Example 5-28 Creating a threshold violations event monitor file

```
$ db2 "create event monitor wlm_thresh for threshold violations write to
file '/evmon/wlm/thresh'"
DB20000I The SQL command completed successfully.
$ db2 "create event monitor wlm_act for activities write to file
'/evmon/wlm/activities' autostart"
DB20000I The SQL command completed successfully.

$ db2 commit
DB20000I The SQL command completed successfully.
```

2. Activate the event monitor:

Use the SET EVENT MONITOR STATE statement to activate the event monitor.

Example 5-29 shows how to set the event monitor active.

Example 5-29 Setting the event monitor active

```
$ db2 set event monitor wlm_thresh state 1
DB20000I The SQL command completed successfully.
```

```
$ db2 set event monitor wlm_act state 1
DB20000I The SQL command completed successfully.
```

```
$ db2 commit
DB20000I The SQL command completed successfully.
```

3. Create a threshold object:

Example 5-30 shows how to create a threshold and specify the COLLECT ACTIVITY DATA keyword.

Example 5-30 Using the CREATE THRESHOLD command

```
$ db2 "create threshold rowsreturnthresh for database activities
enforcement database when SQLROWSRETURNED > 10000 collect activity data
without details stop execution"
DB20000I The SQL command completed successfully.
```

4. Run your query.

We run an SQL statement that should return over 10000 rows. Due to the threshold restriction, our query was stopped after received 10000 rows. The SQL code SQL4712N with SQLSTATE 5U026 was also returned.

5. Format the threshold violations event monitor file and see the output:

Example 5-31 shows the command and output of formatting the threshold violations event monitor.

Example 5-31 Output of formatting the threshold violations event monitor

```
$ db2evmon -path /evmon/wlm/thresh > db2evmon1.out
$ more db2evmon1.out
<<extract from db2evmon1.out>>
5) Threshold Violation ...
  Threshold ID      : 1
  Activity ID       : 1
  Appl Handle       : 337
  Application ID    : *LOCAL.DB2_01.070821181355
  UOW ID            : 1
  Coordinating Partition : 0
```

```

Time of Violation      : 2007-08-21 11:14:21.000000
Threshold Max Value   : 10000
Threshold Queue Size  : 0
Activity Collected?  : Yes
Threshold Predicate    : SQLRowsReturned
Threshold Action      : Stop

```

6. Format the activity event monitor file and see the output:

Example 5-32 shows the command and output of formatting the activity event monitor. In our example, we only collect default activity information that includes SQLCODE, SQLSTATE and reason code. If you want to see which query was violated by the threshold, you need to set COLLECT ACTIVITY DATA WITH DETAILS or COLLECT ACTIVITY DATA WITH DETAILS AND VALUES for the threshold.

Example 5-32 Output of formatting the activity event monitor

```

$ db2evmon -path /evmon/wlm/act > db2evmon2.out
$ more db2evmon2.out
<<extract from db2evmon2.out>>
31) Activity ...
   Activity ID                : 1
   Activity Secondary ID      : 0
   Appl Handle                 : 337
   UOW ID                      : 1
   Service Superclass Name    : SYSDEFAULTUSERCLASS
   Service Subclass Name      : SYSDEFAULTSUBCLASS

   Activity Type              : READ_DML
   Parent Activity ID         : 0
   Parent UOW ID              : 0
   Coordinating Partition     : 0
   Workload ID                 : 1
   Workload Occurrence ID     : 2
   Database Work Action Set ID : 0
   Database Work Class ID     : 0
   Service Class Work Action Set ID : 0
   Service Class Work Class ID : 0
   Time Created                : 2007-08-21 11:15:01.750519
   Time Started                : 2007-08-21 11:15:01.750557
   Time Completed              : 2007-08-21 11:16:53.405745
   Activity captured while in progress: FALSE

   Application ID              : *LOCAL.DB2_01.070821181355
   Application Name             : db2bp.exe
   Session Auth ID             : db2inst1
   Client Userid                :

```



```

Client Workstation Name      :
Client Applname             :
Client Accounting String    :
SQLCA:
  SQL4712N The threshold "ROWSRETURNTHRESH" has been exceeded. Reason
code = "8".  SQLSTATE=5U026

Query Cost Estimate         : 6399
Query Card Estimate         : 3869893
Execution time               : 0.037908 seconds
Rows Returned                : 10000

```

5.2.3 Statistics event monitor

This section introduces how to use the statistics event monitor and histogram. Compared to the statistics table function, the statistics event monitor can collect information in different level of details based on which COLLECT AGGREGATE keywords is specified. Compared to statement or activities event monitors, the statistics event monitor is an inexpensive method of capturing historical information because the statistics event monitor deals with total activity information instead of individual activities.

The following are the features of statistics event monitor:

- ▶ Which WLM objects can I collect statistics information for?
You can collect information for service classes, work classes, workloads and threshold queues.
- ▶ When is the information useful?
The information is useful for historical analysis.
- ▶ How do I collect the information?
Some statistics are always collected for each object. If you want to collect other statistics, you need to specify the COLLECT AGGREGATE option for the service subclass or for a work action applied to the work class.
- ▶ When is the information collected?
The information is collected when
 - Regular interval that sets the WLM_COLLECT_INT database configuration parameter comes.
 - The WLM_COLLECT_STATS stored procedure is called.

The following monitor elements for statistics are collected for each workload management object:

- ▶ Statistics collected by default
 - Service subclass:
 - coord_act_completed_total
 - coord_act_rejected_total
 - coord_act_aborted_total
 - concurrent_act_top
 - Service superclass:
 - concurrent_connection_top
 - Workload:
 - concurrent_wlo_top
 - concurrent_act_top
 - coord_act_completed_total
 - coord_act_rejected_total
 - coord_act_aborted_total
 - wlo_completed_total
 - Work class (through a work action):
 - act_total
- ▶ Statistics collected when you specify COLLECT AGGREGATE ACTIVITY DATA BASE
 - Service subclass:
 - cost_estimate_top
 - rows_returned_top
 - temp_tablespace_top
 - coord_act_lifetime_top
 - coord_act_lifetime_avg
 - coord_act_exec_time_avg
 - coord_act_queue_time_avg
 - activity lifetime histogram
 - activity execution time histogram
 - activity queue time histogram
 - Work class (through a work action):
 - cost_estimate_top
 - rows_returned_top
 - temp_tablespace_top
 - coord_act_lifetime_top
 - coord_act_lifetime_avg
 - coord_act_exec_time_avg
 - coord_act_queue_time_avg
 - activity lifetime histogram

- activity execution time histogram
 - activity queue time histogram
- ▶ Statistics collected when you specify COLLECT AGGREGATE ACTIVITY DATA EXTENDED
- Service subclass:
 - coord_act_est_cost_avg
 - coord_act_interarrival_time_avg
 - activity inter-arrival time histogram
 - activity estimated cost histogram
 - Work class (through a work action):
 - coord_act_est_cost_avg
 - coord_act_interarrival_time_avg
 - activity inter-arrival time histogram
 - activity estimated cost histogram
- ▶ Statistics collected when you specify COLLECT AGGREGATE REQUEST DATA BASE
- Service subclass:
 - request_exec_time_avg
 - request execution time histogram

For aggregate activity statistics, if COLLECT AGGREGATE ACTIVITY DATA EXTENDED is specified, all the BASE aggregate activity statistics are also collected. If COLLECT AGGREGATE REQUEST DATA BASE is specified, the BASE and EXTENDED aggregate activity statistics are not collected.

If you want to specify both COLLECT AGGREGATE ACTIVITY keyword and COLLECT AGGREGATE REQUEST DATA keyword, you can alter a service subclass to add option. Example 5-33 shows how to specify both keywords. In the first statement result, we know that the PROD_QRY service class already specify the COLLECT AGGREGATE ACTIVITY DATA EXTENDED keyword. After altering to add the COLLECT AGGREGATE REQUEST DATA BASE keyword, the COLLECTAGGREQDATA value is changed to B.

Example 5-33 Commands to alter the service subclass

```
>db2 "SELECT substr(serviceclassname,1,19) as serviceclass_name,
collectaggactdata, collectaggreqdata FROM syscat.serviceclasses WHERE
serviceclassname='PROD_QRY'"
```

SERVICECLASS_NAME	COLLECTAGGACTDATA	COLLECTAGGREQDATA
PROD_QRY	E	N

1 record(s) selected.

```
>db2 "alter service class "PROD_QRY" under "HIGHLVL" collect aggregate request
data base"
```

DB20000I The SQL command completed successfully.

```
>db2 "SELECT substr(serviceclassname,1,19) as superclass_name,
collectaggactdata, collectaggreqdata FROM syscat.serviceclasses WHERE
serviceclassname='PROD_QRY'"
```

SUPERCLASS_NAME	COLLECTAGGACTDATA	COLLECTAGGREQDATA
PROD_QRY	E	B

1 record(s) selected.

Resetting statistics on workload management objects

There are four events that can reset statistics:

- ▶ The WLM_COLLECT_STATS stored procedure is called.
The stored procedure collects the current values of the in-memory statistics and reset the statistics. If one user calls the stored procedure, a reset of the workload manager statistics applies to all users.
- ▶ The periodic workload management statistics collection and reset process controlled by the WLM_COLLECT_INT database configuration parameter causes a collection and reset.
WLM_COLLECT_INT enables event monitor to capture statistics automatically at regular intervals.
- ▶ The database is reactivated.
Every time the database is activated on a database partition, the statistics for all workload management objects on that database partition are reset.
- ▶ The object for which the statistics are maintained is modified and the change is committed.
For example if a service subclass is altered, when the ALTER is committed, the in-memory statistics for that service subclass are reset.

Resetting statistics also applied to the statistics table functions. When you use the statistics table functions or the statistics event monitor, you check timestamp the statistics were reset.

Automatic collecting statistics in a specific time frame

Use the following steps to automatically collect statistics for a given workload management object in a given time frame:

1. Create and activate a statistics event monitor:

Use the CREATE EVENT MONITOR statement to create a STATISTICS event monitor as shown in Example 5-34. This event monitor writes data to file.

Example 5-34 Creating a statistics event monitor file

```
$ db2 "create event monitor wlm_stats for statistics write to file
'/evmon/wlm/stats'"
DB20000I The SQL command completed successfully.
```

Use the SET EVENT MONITOR STATE statement to activate the event monitor as shown in Example 5-35.

Example 5-35 Setting the event monitor active

```
$ db2 set event monitor wlm_stats state 1
DB20000I The SQL command completed successfully.
```

2. Enable the collection of additional statistics:

By default, only a minimal set of statistics is collected for each workload management object.

Example 5-36 shows how to alter the default service class to specify the COLLECT AGGREGATE ACTIVITY DATA BASE keyword.

Example 5-36 Using the ALTER SERVICE CLASS

```
$ db2 "alter service class SYSDEFAULTSUBCLASS under SYSDEFAULTUSERCLASS
COLLECT AGGREGATE ACTIVITY DATA BASE"
DB20000I The SQL command completed successfully.
```

3. Specify a collection interval:

Update the database configuration parameter WLM_COLLECT_INT. Example 5-37 shows how to update the database configuration parameter. In this case, statistics information is sent to the statistics event monitor every 5 minutes. After you perform the preceding steps, workload management statistics are written to the statistics event monitor every wlm_collect_int minutes.

Example 5-37 Updating the wlm_collect_int database configuration parameter

```
>db2 get db cfg for wlmdb | grep 'WLM_COLLECT_INT'
WLM Collection Interval (minutes)      (WLM_COLLECT_INT) = 0
```

```
>db2 update db cfg for sample using wlm_collect_int 5 immediate
update db cfg for wlmdb using wlm_collect_int 5 immediate
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
```

```
>db2 get db cfg for sample | grep 'WLM_COLLECT_INT'
WLM Collection Interval (minutes) (WLM_COLLECT_INT) = 5
```

4. Run your activities.
5. Formats the statistics event monitor file:

Example 5-38 shows the formatted output of the statistics event monitor. The statistics collecting interval is 5 minutes, from TIME OF LAST RESET(2007-08-22 09:21:36.238396) to STATISTICS TIMESTAMP(2007-08-22 09:16:36.231024).

To see the value of Coordinator Activity Estimated Cost Average, Coordinator Activity Interarrival Time Average, you need to specify COLLECT AGGREGATE ACTIVITY DATA EXTENDED option for the service subclass. To see the value of Request Execution Time Average, you need to specify COLLECT AGGREGATE REQUEST DATA BASE option for the service subclass.

Example 5-38 Output of formatting the statistics event monitor

```
647) Service Class Statistics ...
Service Superclass Name      : SYSDEFAULTUSERCLASS
Service Subclass Name       : SYSDEFAULTSUBCLASS
Service Class ID            : 13
Temp Tablespace high water mark : 0
Rows Returned high water mark : 262
Cost Estimate high water mark : 1001074
Coordinator Completed Activity Count : 78
Coordinator Aborted Activity Count : 0
Coordinator Rejected Activity Count : 0
Coordinator Activity Lifetime high water mark : 79
Coordinator Activity Lifetime Average : 12
Coordinator Activity Queue Time Average : 0
Coordinator Activity Execution Time Average : 1
Coordinator Activity Estimated Cost Average : -1
Coordinator Activity Interarrival Time Average: -1
Request Execution Time Average : -1
Number Concurrent Activities high water mark : 1
Number Concurrent Connections high water mark : 4
Statistics Timestamp       : 2007-08-22 09:21:36.238396
Time of Last Reset        : 2007-08-22 09:16:36.231024
```

Histograms in workload management

DB2 workload management histograms are useful for analyzing overall activity behaviors in a DB2 system. The histograms information can be collected and sent to a statistics event monitor when you specify COLLECT AGGREGATE keywords for service subclasses or work classes (through work actions).

Every DB2 WLM histogram has 41 bins for the data collected. Each bin has the range from top to bottom. You can see the number (frequency) of the activities or requests within the range. Example 5-39 shows the histogram section of a statistics event monitor output. Three activities (Number in bin) had an execution time in the range zero (Bottom) milliseconds to one (Top) milliseconds.

Example 5-39 Extract histogram output from statistics event monitor file

```
49) Histogram Bin ...
   Top                : 1
   Bottom             : 0
   Number in bin      : 3
   Bin ID             : 1
   Service Class ID   : 13
   Work Action Set ID : 0
   Work Class ID      : 0
   Statistics Timestamp : 2007-08-22 14:09:05.488364
   Histogram Type     : CoordActExecTime
```

There are six types of WLM histogram:

- ▶ **Coordinator activity queue time (CoordActQueueTime)**
 This is a histogram of the queue times (the amount of time that an activity spends in threshold queues before it starts executing).
 You can obtain this type of histogram by specifying AGGREGATE ACTIVITY DATA BASE or AGGREGATE ACTIVITY DATA EXTENDED for the service subclass or for a work action applied to the work class.
- ▶ **Coordinator activity execution time (CoordActExecTime)**
 This is a histogram of the time a non-nested activity spends executing at the coordinator partition that does not include time spent queued by DB2 or idle time.
 You can obtain this type of histogram by specifying AGGREGATE ACTIVITY DATA BASE or AGGREGATE ACTIVITY DATA EXTENDED for a service subclass or for a work action applied to the work class.
- ▶ **Coordinator activity life time (CoordActLifetime)**
 This is a histogram of the elapsed lifetime of activities, measured from the time when an activity enters the system until the activity completes execution.

You can obtain this type of histogram by specifying AGGREGATE ACTIVITY DATA BASE or AGGREGATE ACTIVITY DATA EXTENDED for a service subclass or for a work action applied to the work class.

- ▶ Coordinator activity inter-arrival time (CoordActInterArrivalTime)

This is a histogram of the time interval between the arrival of one activity and the arrival of the next activity.

You can obtain this type of histogram by specifying AGGREGATE ACTIVITY DATA EXTENDED for a service subclass or for a work action applied to the work class.

- ▶ Coordinator activity estimated cost (CoordActEstCost)

This is a histogram of the estimated cost of non-nested DML activities.

You can obtain this type of histogram by specifying specify AGGREGATE ACTIVITY DATA EXTENDED for a service subclass or for a work action applied to the work class.

- ▶ Request execution time (ReqExecTime)

This is a histogram of the time a request spends executing. The execution time for requests is collected in a histogram for each database partition and for all requests.

You can obtain this type of histogram by specifying AGGREGATE REQUEST DATA BASE for a service subclass.

The difference in histogram between AGGREGATE ACTIVITY DATA BASE and DATA EXTENDED is that DATA EXTENDED can collect more histogram types than DATA BASE:

- ▶ COLLECT AGGREGATE ACTIVITY DATA BASE
 - CoordActQueueTime
 - CoordActExecTime
 - CoordActLifeTime
- ▶ COLLECT AGGREGATE ACTIVITY DATA EXTENDED
 - CoordActQueueTime
 - CoordActExecTime
 - CoordActLifeTime
 - CoordActInterArrivalTime
 - CoordActEstCost

Histogram template defines the range. The default template is SYSDEFAULTHISTOGRAM. Example 5-40 shows the range of each bins for the SYSDEFAULTHISTOGRAM. The output shows that the high bin value is 21600000 milliseconds. If you want to increase or decrease the high bin value of the histogram, you can use ALTER HISTOGRAM TEMPLATE statement or

CREATE HISTOGRAM TEMPLATE statement to create your histogram template.

Example 5-40 The range of SYSDEFAULTHISTOGRAM

```
>db2 "SELECT binid,binuppervalue FROM syscat.histogramtemplatebins WHERE
templatenam='SYSDEFAULTHISTOGRAM' ORDER BY binuppervalue"
```

BINID	BINUPPERVALUE
1	1
2	2
3	3
4	5
5	8
6	12
7	19
8	29
9	44
10	68
11	103
12	158
13	241
14	369
15	562
16	858
17	1309
18	1997
19	3046
20	4647
21	7089
22	10813
23	16493
24	25157
25	38373
26	58532
27	89280
28	136181
29	207720
30	316840
31	483283
32	737162
33	1124409
34	1715085
35	2616055
36	3990325
37	6086529
38	9283913
39	14160950
40	21600000

40 record(s) selected.

You can follow these steps to use histograms for a service class:

1. Create and activate an event monitor:

Use the CREATE EVENT MONITOR statement to create a statistics event monitor. Example 5-41 shows how to create a statistics event monitor that writes data to a table.

Example 5-41 Creating a statistics event monitor table

```
>db2 "create event monitor db2statistics for statistics write to table
scstats (table scstats_db2statistics in userspace1), wcstats (table
wcstats_db2statistics in userspace1), wlstats (table wlstats_db2statistics
in userspace1 pctdeactivate 100), qstats (table qstats_db2statistics in
userspace1), histogrambin (table histogrambin_db2statistics in userspace1),
control (table control_db2statistics in userspace1)"
DB20000I The SQL command completed successfully.
```

Use the SET EVENT MONITOR STATE statement to activate the event monitor. Example 5-42 shows how to set the event monitor active.

Example 5-42 Setting the event monitor to active

```
>db2 "set event monitor db2statistics state 1"
DB20000I The SQL command completed successfully.
```

2. Enable the collection of histograms for the service subclass:

Example 5-43 shows how to alter the default service class to specify the COLLECT AGGREGATE ACTIVITY keyword. The COLLECT AGGREGATE ACTIVITY DATA BASE option on the service class produces the histogram types coordinator activity life time, coordinator activity execution time, and coordinator activity queue time.

Example 5-43 Using the ALTER SERVICE CLASS

```
>db2 "alter service class sysdefaultsubclass under sysdefaultuserclass
COLLECT AGGREGATE ACTIVITY DATA BASE"
DB20000I The SQL command completed successfully.
```

3. Create a view to make querying the HISTOGRAMBIN_DB2STATISTICS table easier.

Example 5-44 shows how to create the view for histograms. This view returns histogram data across multiple intervals to produce a single histogram of a given service class.

Example 5-44 Creating the view of histograms

```
>db2 "create view histograms (histogram_type, service_superclass,
service_subclass, bin_top, number_in_bin) as select distinct
substr(histogram_type,1,24) as histogram_type,
substr(parentserviceclassname,1,24) as service_superclass, sub
str(serviceclassname,1,24) as service_subclass, top as bin_top,
sum(number_in_bin) as number_in_bin from histogrambin_db2statistics h,
syscat.serviceclasses s where h.service_class_id = s.serviceclassid group
by histogram_type, parentserviceclassname, serviceclassname, top"
DB20000I The SQL command completed successfully.
```

4. Run your activities.

After the activities have finished, the WLM_COLLECT_STATS stored procedure is called. In this case, WLM_COLLECT_INT database configuration parameter sets 0. Some activities are run twice and the stored procedure is called twice.

5. Look the statistics:

Example 5-45 shows how many rows are stored in the HISTOGRAMBIN_DB2STATISTICS table. In this case, we collected 246 rows in the HISTOGRAMBIN_DB2STATISTICS table:

246 rows = 3(types of histogram) x 2(times information is collected) x 41(bins)

Example 5-45 Counting the HISTOGRAMBIN_DB2STATISTICS table

```
>db2 "SELECT count(*) as count FROM histogrambin_db2statistics"
```

```
COUNT
-----
      246
```

```
1 record(s) selected.
```

Example 5-46 shows the CoordActExecTime histogram for SYSDEFAULTUSERCLASS. The BIN_TOP value -1 means infinity. If any activity runs over 21600000 milliseconds, the NUMBER_IN_BIN will not be 0. Six activities had the execution time between 0 and 1 milliseconds.

Example 5-46 Analyses of the CoordActExecTime for SYSDEFAULTUSERCLASS

```
>db2 "SELECT bin_top, number_in_bin FROM histograms WHERE
      histogram_type='CoordActExecTime' and
      service_superuserclass='SYSDEFAULTUSERCLASS' ORDER BY bin_top"
```

BIN_TOP	NUMBER_IN_BIN
-1	0
1	6
2	2
3	0
5	0
8	0
12	0
19	0
29	0
44	0
68	0
103	1
158	0
241	0
369	1
562	0
858	0

1309	0
1997	0
3046	0
4647	1
7089	0
10813	0
16493	0
25157	0
38373	0
58532	0
89280	0
136181	0
207720	0
316840	0
483283	0
737162	0
1124409	0
1715085	0
2616055	0
3990325	0
6086529	0
9283913	0
14160950	0
21600000	0

41 record(s) selected.

Example 5-47 shows the CoordActLifetime histogram for SYSDEFAULTUSERCLASS. Five activities are counted in the lowest bin. One activity has the largest life time between 3046 and 4647 milliseconds.

Example 5-47 Analysis of the CoordActLifetime for SYSDEFAULTUSERCLASS

```
>db2 "SELECT bin_top, number_in_bin FROM histograms WHERE
      histogram_type='CoordActLifetime' and
      service_superclass='SYSDEFAULTUSERCLASS' ORDER BY bin_top"
```

BIN_TOP	NUMBER_IN_BIN
-1	0
1	5
2	1
3	0
5	0
8	0
12	0
19	0
29	0
44	0
68	0
103	3
158	0
241	0
369	1
562	0
858	0

1309	0
1997	0
3046	0
4647	1
7089	0
10813	0
16493	0
25157	0
38373	0
58532	0
89280	0
136181	0
207720	0
316840	0
483283	0
737162	0
1124409	0
1715085	0
2616055	0
3990325	0
6086529	0
9283913	0
14160950	0
21600000	0

41 record(s) selected.

Example 5-48 shows the CoordActQueueTime histogram for SYSDEFAULTUSERCLASS. All 11 activities are counted in the lowest bin because nothing is queued in this example.

Example 5-48 Analysis for the CoordActQueueTime for SYSDEFAULTUSERCLASS

```
>db2 "SELECT bin_top, number_in_bin FROM histograms WHERE
histogram_type='CoordActQueueTime' and service_superclass='SYSD
EFAULTUSERCLASS' ORDER BY bin_top"
```

BIN_TOP	NUMBER_IN_BIN
-1	0
1	11
2	0
3	0
5	0
8	0
12	0
19	0
29	0
44	0
68	0
103	0
158	0
241	0
369	0

562	0
858	0
1309	0
1997	0
3046	0
4647	0
7089	0
10813	0
16493	0
25157	0
38373	0
58532	0
89280	0
136181	0
207720	0
316840	0
483283	0
737162	0
1124409	0
1715085	0
2616055	0
3990325	0
6086529	0
9283913	0
14160950	0
21600000	0

41 record(s) selected.

5.3 Workload profiling and capturing

DB2 WLM provides you the capability to capture detailed workload profiles and performance information in the database system for refining the workload and service class definitions to achieve the business objectives. In this section, we use the simple WLM configuration customized in Chapter 4, “Customizing the WLM execution environments” on page 61 to demonstrate what data you can capture and how to capture them. We provide the monitoring examples and results.

Table 5-3 shows the simple WLM configuration from Chapter 4, “Customizing the WLM execution environments” on page 61.

Table 5-3 WLMDB WLM configuration

Service superclass	Service subclass	Subclass action (collect information)	Workload	Workload Identification
HIGHLVL	ADMINS	AGGREGATE REQUEST	WL_ADMIN	SESSION_USER GROUP: DB2ADM
HIGHLVL	BATCH	AGGREGATE REQUEST	WL_BATCH	CLIENT_USERID: BATCH
HIGHLVL	PROD_RPT	AGGREGATE ACTIVITY EXTENDED AGGREGATE REQUEST	WL_PROD_RPT	APPLNAME: dss.exe

Service superclass	Service subclass	Subclass action (collect information)	Workload	Workload Identification
HIGHLVL	PROD_QRY	AGGREGATE ACTIVITY EXTENDED AGGREGATE REQUEST	WL_PROD_QRY	SESSION_USER GROUP: DSSGROUP

We have the monitor environment on our system as follows:

- ▶ Database configuration parameter WLM_COLLECT_INT is set to 5
- ▶ Statistics event monitor BASIC_MON is created and activated

Limited by the available monitoring time of 90 minutes, we reduce the interval to capture shorter term behavior. The statistics information is captured and sent to the statistics event monitor at 5 minutes interval automatically.

5.3.1 Monitoring overall database system behavior

WLM table functions are useful in capturing the activities taking place in a database system. In this section, we demonstrate how to use the WLM table functions to observe work being performed in the system. We provides SQL statements to list workload occurrences, agents, activities associated within a workload, and summary statistics.

List workload occurrences at the service subclass level

You can use table function

WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES to list workload occurrences at the service subclass level.

Example 5-49 shows SQL statement and the output. We observed that all workload occurrences are collected at the coordinator partition 0. This means that all applications connect to the coordinator partition 0.

Example 5-49 List of workload occurrences at the service subclass level

```
>db2 "SELECT substr(service_superclass_name,1,8) as superclass_name,
substr(service_subclass_name,1,10) as subclass_name,
substr(char(dbpartitionnum),1,6) as part, substr(workload_name,1,12) as
workload_name, substr(char(application_handle),1,4) as applhandle,
workload_occurrence_state FROM
_WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('HIGHLVL','ADMINS',-2)) order
by superclass_name, subclass_name, dbpartitionnum, workload_name"
```

SUPERCLASS_NAME	SUBCLASS_NAME	PART	WORKLOAD_NAME	APPLHANDLE	WORKLOAD_OCCURRENCE_STATE
HIGHLVL	ADMINS	0	WL_ADMIN	4908	UOWWAIT
HIGHLVL	ADMINS	0	WL_ADMIN	4925	UOWWAIT
HIGHLVL	ADMINS	0	WL_ADMIN	4961	UOWWAIT

```
HIGHLVL      ADMINS      0      WL_ADMIN      4989      UOWEXEC
```

4 record(s) selected.

```
>db2 "SELECT substr(service_superclass_name,1,8) as superclass_name,
substr(service_subclass_name,1,10) as subclass_name,
substr(char(dbpartitionnum),1,6) as part, substr(workload_name,1,12) as
workload_name, substr(char(application_handle),1,4) as applhandle,
workload_occurrence_state FROM
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('HIGHLVL','BATCH',-2)) order
by superclass_name, subclass_name, dbpartitionnum, workload_name"
```

SUPERCLASS_NAME	SUBCLASS_NAME	PART	WORKLOAD_NAME	APPLHANDLE	WORKLOAD_OCCURRENCE_STATE
HIGHLVL	BATCH	0	WL_BATCH	4929	UOWEXEC
HIGHLVL	BATCH	0	WL_BATCH	4931	UOWEXEC
HIGHLVL	BATCH	0	WL_BATCH	4982	UOWWAIT

3 record(s) selected.

```
>db2 "SELECT substr(service_superclass_name,1,8) as superclass_name,
substr(service_subclass_name,1,10) as subclass_name,
substr(char(dbpartitionnum),1,6) as part, substr(workload_name,1,12) as
workload_name, substr(char(application_handle),1,4) as applhandle,
workload_occurrence_state FROM
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('HIGHLVL','PROD_RPT',-2))
order by superclass_name, subclass_name, dbpartitionnum, workload_name"
```

SUPERCLASS_NAME	SUBCLASS_NAME	PART	WORKLOAD_NAME	APPLHANDLE	WORKLOAD_OCCURRENCE_STATE
HIGHLVL	PROD_RPT	0	WL_PROD_RPT	4946	UOWEXEC
HIGHLVL	PROD_RPT	0	WL_PROD_RPT	4947	UOWEXEC
HIGHLVL	PROD_RPT	0	WL_PROD_RPT	4951	UOWEXEC

3 record(s) selected.

```
>db2 "SELECT substr(service_superclass_name,1,8) as superclass_name,
substr(service_subclass_name,1,10) as subclass_name,
substr(char(dbpartitionnum),1,6) as part, substr(workload_name,1,12) as
workload_name, substr(char(application_handle),1,4) as applhandle,
workload_occurrence_state FROM
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('HIGHLVL','PROD_QRY',-2))
order by superclass_name, subclass_name, dbpartitionnum, workload_name"
```

SUPERCLASS_NAME	SUBCLASS_NAME	PART	WORKLOAD_NAME	APPLHANDLE	WORKLOAD_OCCURRENCE_STATE
HIGHLVL	PROD_QRY	0	WL_PROD_QRY	4937	UOWEXEC
HIGHLVL	PROD_QRY	0	WL_PROD_QRY	4938	UOWEXEC
HIGHLVL	PROD_QRY	0	WL_PROD_QRY	4969	UOWEXEC
HIGHLVL	PROD_QRY	0	WL_PROD_QRY	4971	UOWEXEC
HIGHLVL	PROD_QRY	0	WL_PROD_QRY	4972	UOWEXEC
HIGHLVL	PROD_QRY	0	WL_PROD_QRY	4970	UOWEXEC
HIGHLVL	PROD_QRY	0	WL_PROD_QRY	4973	UOWEXEC
HIGHLVL	PROD_QRY	0	WL_PROD_QRY	4974	UOWEXEC
HIGHLVL	PROD_QRY	0	WL_PROD_QRY	4975	UOWEXEC
HIGHLVL	PROD_QRY	0	WL_PROD_QRY	4976	UOWEXEC

10 record(s) selected.

List the agents working in the database

You can use table function WLM_GET_SERVICE_CLASS_AGENTS to list the agents working in the database.

Example 5-50 shows the SQL statements and the output. In the first output, we can see the number of agents for each service subclasses. For example, 72 agents are assigned to PROD_QRY service subclass.

Using the second query, you can see the details of the agents. For example, the coordinator agent distributes database requests to subagents across partitions. The agents which have SUBSECTION value in REQTYPE are subagents. There are total 17 agents performed in the application handle 4931.

Example 5-50 Output of WLM_GET_SERVICE_CLASS_AGENTS table function

```
>db2 "SELECT substr(agents.service_superclass_name,1,19) as superclass_name,
substr(agents.service_subclass_name,1,19) as subclass_name, count(*) as
agent_count FROM
TABLE(WLM_GET_SERVICE_CLASS_AGENTS(' ', ' ', CAST(NULL AS BIGINT), -2)) as agents
WHERE agent_state = 'ACTIVE' GROUP BY service_superclass_name,
service_subclass_name ORDER BY service_superclass_name, service_subclass_name"
```

SUPERCLASS_NAME	SUBCLASS_NAME	AGENT_COUNT
HIGHLVL	ADMINS	5
HIGHLVL	BATCH	25
HIGHLVL	PROD_QRY	72
HIGHLVL	PROD_RPT	42
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	7
-	-	3

6 record(s) selected.

```
>db2 "SELECT substr(service_superclass_name,1,19) as superclass_name,
substr(service_subclass_name,1,19) as subclass_name,
substr(workload_name,1,11) as workload_name,
substr(char(application_handle),1,7) as applhandle,
substr(char(dbpartitionnum),1,4) as part, substr(agent_state,1,10) as
agentstate, substr(event_state,1,10) as eventstate, substr(request_type,1,7) as
reqtype, substr(char(uow_id),1,6) as uow_id, substr(char(activity_id),1,6) as
act_id FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS(' ', ' ', CAST(NULL AS BIGINT),
-2)) WHERE agent_type <> 'OTHER' and request_type <> 'INTERNAL0' ORDER BY
applhandle, part, agent_tid"
```

<<extract of output>>

SUPERCLASS_NAME	SUBCLASS_NAME	WORKLOAD_NAME	APPLHANDLE	PART	AGENTSTATE	EVENTSTATE	REQTYPE
UOW_ID	ACT_ID						

-	-	WL_BATCH	4931	0	ASSOCIATED	IDLE	INTERNA
17	1						
HIGHLVL	BATCH	WL_BATCH	4931	0	ACTIVE	IDLE	OPEN
17	1						
HIGHLVL	BATCH	-	4931	1	ACTIVE	EXECUTING	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	1	ACTIVE	IDLE	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	1	ACTIVE	EXECUTING	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	1	ACTIVE	EXECUTING	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	2	ACTIVE	EXECUTING	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	2	ACTIVE	IDLE	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	2	ACTIVE	EXECUTING	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	3	ACTIVE	EXECUTING	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	3	ACTIVE	IDLE	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	3	ACTIVE	EXECUTING	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	3	ACTIVE	EXECUTING	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	4	ACTIVE	EXECUTING	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	4	ACTIVE	IDLE	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	4	ACTIVE	EXECUTING	SUBSECT
17	1						
HIGHLVL	BATCH	-	4931	4	ACTIVE	EXECUTING	SUBSECT
17	1						
HIGHLVL	PROD_RPT	WL_PROD_RPT	4951	0	ACTIVE	IDLE	OPEN
19	1						
HIGHLVL	PROD_RPT	-	4951	1	ACTIVE	EXECUTING	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	1	ACTIVE	IDLE	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	1	ACTIVE	EXECUTING	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	1	ACTIVE	EXECUTING	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	2	ACTIVE	EXECUTING	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	2	ACTIVE	IDLE	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	2	ACTIVE	EXECUTING	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	3	ACTIVE	EXECUTING	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	3	ACTIVE	IDLE	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	3	ACTIVE	EXECUTING	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	3	ACTIVE	EXECUTING	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	4	ACTIVE	EXECUTING	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	4	ACTIVE	IDLE	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	4	ACTIVE	EXECUTING	SUBSECT
19	1						
HIGHLVL	PROD_RPT	-	4951	4	ACTIVE	EXECUTING	SUBSECT
19	1						

HIGHLVL 3	PROD_QRY	WL_PROD_QRY	4970	0	ACTIVE	IDLE	OPEN	1
HIGHLVL 3	PROD_QRY	-	4970	1	ACTIVE	IDLE	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4970	1	ACTIVE	EXECUTING	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4970	2	ACTIVE	EXECUTING	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4970	2	ACTIVE	IDLE	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4970	3	ACTIVE	IDLE	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4970	3	ACTIVE	EXECUTING	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4970	4	ACTIVE	IDLE	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4970	4	ACTIVE	EXECUTING	SUBSECT	1
HIGHLVL 3	PROD_QRY	WL_PROD_QRY	4973	0	ACTIVE	IDLE	OPEN	1
HIGHLVL 3	PROD_QRY	-	4973	1	ACTIVE	IDLE	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4973	1	ACTIVE	EXECUTING	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4973	2	ACTIVE	IDLE	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4973	2	ACTIVE	EXECUTING	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4973	3	ACTIVE	IDLE	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4973	3	ACTIVE	EXECUTING	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4973	4	ACTIVE	IDLE	SUBSECT	1
HIGHLVL 3	PROD_QRY	-	4973	4	ACTIVE	EXECUTING	SUBSECT	1

List current activities associated with a workload occurrence

You can use table function

WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES to list current activities associated with a workload occurrence.

Example 5-51 shows the SQL statements and the output. The first SQL statement provides the number of activities for each workloads. For example, 44 activities are assigned to WL_PROD_QRY workload.

In the second output, we can see the type of activity (activity_type) for each activity. Subagents have OTHER in the activity type.

Example 5-51 List the current activities associated with a workload occurrence

```
>db2 "SELECT substr(workload_name,1,22) as workload_name, count(*) as
total_exe_act FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES(' ', ' ',
-2)) as apps,
TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(APPS.APPLICATION_HANDLE, -2)) as
appacts WHERE apps.dbpartitionnum = apps.coord_partition_num and activity_state
```

```
= 'executing' and nesting_level = 0 GROUP BY workload_name ORDER BY
workload_name"
```

WORKLOAD_NAME	TOTAL_EXE_ACT
WL_ADMIN	1
WL_BATCH	10
WL_PROD_QRY	44
WL_PROD_RPT	12

4 record(s) selected.

```
>db2 "SELECT substr(char(application_handle),1,4) as applhandle,
substr(char(dbpartitionnum),1,4) as part, substr(char(uowid),1,5) as uowid,
substr(char(activity_id),1,5) as actid, substr(activity_state,1,9) as actstate,
substr(activity_type,1,9) as acttype FROM
TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(cast(null as bigint), -2)) ORDER
BY applhandle, part, uowid, actid"
```

<<extract of output>>

APPLHANDLE	PART	UOWID	ACTID	ACTSTATE	ACTTYPE
4931	0	17	1	EXECUTING	READ_DML
4951	0	19	1	EXECUTING	READ_DML
4970	0	1	3	EXECUTING	READ_DML
4973	0	1	3	EXECUTING	READ_DML
4931	1	17	1	EXECUTING	OTHER
4951	1	19	1	EXECUTING	OTHER
4970	1	1	3	EXECUTING	OTHER
4973	1	1	3	EXECUTING	OTHER
4931	2	17	1	EXECUTING	OTHER
4951	2	19	1	EXECUTING	OTHER
4970	2	1	3	EXECUTING	OTHER
4973	2	1	3	EXECUTING	OTHER
4931	3	17	1	EXECUTING	OTHER
4951	3	19	1	EXECUTING	OTHER
4970	3	1	3	EXECUTING	OTHER
4973	3	1	3	EXECUTING	OTHER
4931	4	17	1	EXECUTING	OTHER
4951	4	19	1	EXECUTING	OTHER
4970	4	1	3	EXECUTING	OTHER
4973	4	1	3	EXECUTING	OTHER

Obtain summary statistics across partitions at the service superclass level

You can use table function `WLM_GET_SERVICE_SUPERCLASS_STATS` to obtain summary statistics across partitions at the service superclass level.

Example 5-52 shows the SQL statements and the output. In this case, 21 is the highest number of concurrent coordinator connections that has been reached in `HIGHLVL` service superclass since the last reset.

Example 5-52 Output of `WLM_GET_SERVICE_SUPERCLASS_STATS` table function

```
>db2 "SELECT substr(service_superclass_name, 1, 26) as superclass_name,
substr(char(dbpartitionnum),1,4) as part, last_reset, concurrent_connection_top
```

```
FROM TABLE(WLM_GET_SERVICE_SUPERCLASS_STATS('', -2)) ORDER BY
service_superclass_name, dbpartitionnum"
```

SUPERCLASS_NAME	PART	LAST_RESET	CONCURRENT_CONNECTION_TOP
HIGHLVL	0	2007-08-30-22.37.08.502709	21
HIGHLVL	1	2007-08-30-22.37.08.502709	0
HIGHLVL	2	2007-08-30-22.37.08.502709	0
HIGHLVL	3	2007-08-30-22.37.08.502709	0
HIGHLVL	4	2007-08-30-22.37.08.502709	0
SYSDEFAULTMAINTENANCECLASS	0	2007-08-30-22.37.08.502709	0
SYSDEFAULTMAINTENANCECLASS	1	2007-08-30-22.37.08.502709	0
SYSDEFAULTMAINTENANCECLASS	2	2007-08-30-22.37.08.502709	0
SYSDEFAULTMAINTENANCECLASS	3	2007-08-30-22.37.08.502709	0
SYSDEFAULTMAINTENANCECLASS	4	2007-08-30-22.37.08.502709	0
SYSDEFAULTSYSTEMCLASS	0	2007-08-30-22.37.08.502709	6
SYSDEFAULTSYSTEMCLASS	1	2007-08-30-22.37.08.502709	1
SYSDEFAULTSYSTEMCLASS	2	2007-08-30-22.37.08.502709	0
SYSDEFAULTSYSTEMCLASS	3	2007-08-30-22.37.08.502709	0
SYSDEFAULTSYSTEMCLASS	4	2007-08-30-22.37.08.502709	0
SYSDEFAULTUSERCLASS	0	2007-08-30-22.37.08.502709	0
SYSDEFAULTUSERCLASS	1	2007-08-30-22.37.08.502709	0
SYSDEFAULTUSERCLASS	2	2007-08-30-22.37.08.502709	0
SYSDEFAULTUSERCLASS	3	2007-08-30-22.37.08.502709	0
SYSDEFAULTUSERCLASS	4	2007-08-30-22.37.08.502709	0

20 record(s) selected.

Obtain summary statistics across partitions at the service subclass level

You can use table function WLM_GET_SERVICE_SUBCLASS_STATS to obtain summary statistics across partitions at the service subclass level.

Example 5-53 shows the SQL statements and the output.

Using the first SQL statements, the summary statistics obtained includes

- ▶ The number of active requests (num_requests_active)
- ▶ Request execution time average (request_exec_time_avg)
- ▶ The number of requests to start executing in this service subclass since the last reset (num_requests_total)
- ▶ Sum of the execution times for requests associated with this service subclass since the last reset (request_exec_time_total) for each service subclasses and partitions

The request_exec_time_avg and the request_exec_time_total divide by 1000 in the statement and get the time on the second time scale.

For example, 25 requests which are associated with PROD_QRY service subclass at partition 2 is the most highest number in the service subclass at all partitions.

Compared to other service subclasses, these requests take much execution time.

In the second output, we can see the high watermark for the concurrency coordinator activities (concurrent_act_top) that run in the service class on each database partition. 10 activities which are associated with PROD_QRY service subclass is the highest number of concurrent activities in the database.

Example 5-53 Summary statistics across partitions at the service subclass level

```
>db2 "SELECT substr(service_superclass_name,1,19) as superclass_name,
substr(service_subclass_name,1,18) as subclass_name,
substr(char(dbpartitionnum),1,4) as part, last_reset, num_requests_active,
cast(request_exec_time_avg / 1000 as decimal(9,3)) as avgreqtime,
num_requests_total, cast(request_exec_time_total / 1000 as decimal(9,3)) as
totalreqtime FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('',' ', -2)) ORDER BY
superclass_name, subclass_name, part"
```

SUPERCLASS_NAME	SUBCLASS_NAME	PART	LAST_RESET	NUM_REQUESTS_ACTIVE
AVGREQTIME	NUM_REQUESTS_TOTAL	TOTALREQTIME		
HIGHLVL	ADMINS	0	2007-08-30-22.37.08.502709	1
0.001	118	0.000		
HIGHLVL	ADMINS	1	2007-08-30-22.37.08.502709	1
0.000	96	0.000		
HIGHLVL	ADMINS	2	2007-08-30-22.37.08.502709	1
0.000	96	0.000		
HIGHLVL	ADMINS	3	2007-08-30-22.37.08.502709	1
0.000	96	0.000		
HIGHLVL	ADMINS	4	2007-08-30-22.37.08.502709	1
0.000	96	0.000		
HIGHLVL	BATCH	0	2007-08-30-22.37.08.502709	3
4.176	25	122.000		
HIGHLVL	BATCH	1	2007-08-30-22.37.08.502709	5
33.043	15	526.000		
HIGHLVL	BATCH	2	2007-08-30-22.37.08.502709	5
34.004	15	522.000		
HIGHLVL	BATCH	3	2007-08-30-22.37.08.502709	6
29.966	14	416.000		
HIGHLVL	BATCH	4	2007-08-30-22.37.08.502709	6
29.333	14	377.000		
HIGHLVL	PROD_QRY	0	2007-08-30-22.37.08.502709	10
64.478	45	2912.000		
HIGHLVL	PROD_QRY	1	2007-08-30-22.37.08.502709	15
11.943	15	175.000		
HIGHLVL	PROD_QRY	2	2007-08-30-22.37.08.502709	15
116.650	25	2928.000		
HIGHLVL	PROD_QRY	3	2007-08-30-22.37.08.502709	17
1.935	13	27.000		
HIGHLVL	PROD_QRY	4	2007-08-30-22.37.08.502709	15
12.120	15	175.000		
HIGHLVL	PROD_RPT	0	2007-08-30-22.37.08.502709	3
12.903	59	727.000		
HIGHLVL	PROD_RPT	1	2007-08-30-22.37.08.502709	10
20.605	40	907.000		
HIGHLVL	PROD_RPT	2	2007-08-30-22.37.08.502709	9
30.617	42	1278.000		
HIGHLVL	PROD_RPT	3	2007-08-30-22.37.08.502709	10
23.004	39	956.000		

HIGHLVL	PROD_RPT	4	2007-08-30-22.37.08.502709	10
22.739	39	954.000		
HIGHLVL	SYSDEFAULTSUBCLASS	0	2007-08-30-22.37.08.502709	0
-	-	-		
HIGHLVL	SYSDEFAULTSUBCLASS	1	2007-08-30-22.37.08.502709	0
-	-	-		
HIGHLVL	SYSDEFAULTSUBCLASS	2	2007-08-30-22.37.08.502709	0
-	-	-		
HIGHLVL	SYSDEFAULTSUBCLASS	3	2007-08-30-22.37.08.502709	0
-	-	-		
HIGHLVL	SYSDEFAULTSUBCLASS	4	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTMAINTENAN	SYSDEFAULTSUBCLASS	0	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTMAINTENAN	SYSDEFAULTSUBCLASS	1	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTMAINTENAN	SYSDEFAULTSUBCLASS	2	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTMAINTENAN	SYSDEFAULTSUBCLASS	3	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTMAINTENAN	SYSDEFAULTSUBCLASS	4	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	0	2007-08-30-22.37.08.502709	6
-	-	-		
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	1	2007-08-30-22.37.08.502709	1
-	-	-		
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	2	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	3	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	4	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	0	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	1	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	2	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	3	2007-08-30-22.37.08.502709	0
-	-	-		
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	4	2007-08-30-22.37.08.502709	0
-	-	-		

40 record(s) selected.

```
>db2 "SELECT substr(service_superclass_name,1,19) as superclass_name,
substr(service_subclass_name,1,18) as subclass_name,
substr(char(dbpartitionnum),1,4) as part, concurrent_act_top as
acthighwatermark FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS(' ', ' ', -2)) ORDER
BY superclass_name, subclass_name, part"
```

SUPERCLASS_NAME	SUBCLASS_NAME	PART	ACTHIGHWATERMARK
HIGHLVL	ADMINS	0	1
HIGHLVL	ADMINS	1	0
HIGHLVL	ADMINS	2	0
HIGHLVL	ADMINS	3	0
HIGHLVL	ADMINS	4	0
HIGHLVL	BATCH	0	3
HIGHLVL	BATCH	1	0
HIGHLVL	BATCH	2	0
HIGHLVL	BATCH	3	0
HIGHLVL	BATCH	4	0
HIGHLVL	PROD_QRY	0	10
HIGHLVL	PROD_QRY	1	0

HIGHLVL	PROD_QRY	2	0
HIGHLVL	PROD_QRY	3	0
HIGHLVL	PROD_QRY	4	0
HIGHLVL	PROD_RPT	0	4
HIGHLVL	PROD_RPT	1	0
HIGHLVL	PROD_RPT	2	0
HIGHLVL	PROD_RPT	3	0
HIGHLVL	PROD_RPT	4	0
HIGHLVL	SYSDEFAULTSUBCLASS	0	0
HIGHLVL	SYSDEFAULTSUBCLASS	1	0
HIGHLVL	SYSDEFAULTSUBCLASS	2	0
HIGHLVL	SYSDEFAULTSUBCLASS	3	0
HIGHLVL	SYSDEFAULTSUBCLASS	4	0
SYSDEFAULTMAINTENAN	SYSDEFAULTSUBCLASS	0	0
SYSDEFAULTMAINTENAN	SYSDEFAULTSUBCLASS	1	0
SYSDEFAULTMAINTENAN	SYSDEFAULTSUBCLASS	2	0
SYSDEFAULTMAINTENAN	SYSDEFAULTSUBCLASS	3	0
SYSDEFAULTMAINTENAN	SYSDEFAULTSUBCLASS	4	0
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	0	0
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	1	0
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	2	0
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	3	0
SYSDEFAULTSYSTEMCLA	SYSDEFAULTSUBCLASS	4	0
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	0	0
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	1	0
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	2	0
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	3	0
SYSDEFAULTUSERCLASS	SYSDEFAULTSUBCLASS	4	0

40 record(s) selected.

Summarizing the execution time and the total activities

We want to know the execution time for each service subclass level. While we have already specified COLLECT AGGREGATE REQUEST DATA BASE option for all service subclasses and created the BASIC_MON statistics event monitor, we can collect the average of request execution time (request_exec_time_avg monitor element).

Example 5-54 shows the view for the statistics event monitor table SCSTATS_BASIC_MON. The view makes it easier to make a graph of the output querying the SCSTATS_BASIC_MON table. We investigate the total coordinator activities completed (coord_act_comp_total) and the average of request execution time (request_exec_time_avg) for each service subclasses during the monitoring term.

The SCSTATSVIEW view shows sum of the COORD_ACT_COMP_TOTAL and REQUEST_EXEC_TIME_AVG from all partitions for each service subclasses at the monitor interval.

Example 5-54 View for the statistics event monitor table

```
CREATE VIEW scstatsview1 (
  statistics_timestamp,
  subclass_name,
  coord_act_comp_total,
```



```

    sum_req_exec_time_avg )
AS
SELECT DISTINCT
  a.statistics_timestamp,
  substr(a.service_subclass_name,1,20),
  CASE WHEN 0 > a.coord_act_completed_total THEN 0
    ELSE a.coord_act_completed_total
  END,
  CASE WHEN 0 > request_exec_time_avg then 0
    ELSE request_exec_time_avg
  END
FROM scstats_basic_mon A;
--
--
CREATE VIEW scstatsview (
  statistics_timestamp,
  subclass_name,
  coord_act_comp_total,
  sum_req_exec_time_avg )
AS
SELECT
  statistics_timestamp,
  subclass_name,
  sum(coord_act_comp_total),
  sum(sum_req_exec_time_avg)
FROM SCSTATSVIEW1
GROUP BY statistics_timestamp, subclass_name
;
```

Example 5-55 shows sample data from the view SCSTATSVIEW.

Example 5-55 Data of SCSTATSVIEW view

```
>db2 "SELECT * FROM scstatsview ORDER BY statistics_timestamp, subclass_name"
```

<<extract of output>>

STATISTICS_TIMESTAMP	SUBCLASS_NAME	COORD_ACT_COMP_TOTAL	SUM_REQ_EXEC_TIME_AVG
2007-08-30-22.03.00.868892	ADMINS	28	15
2007-08-30-22.03.00.868892	BATCH	0	0
2007-08-30-22.03.00.868892	PROD_QRY	0	0
2007-08-30-22.03.00.868892	PROD_RPT	6	274570
2007-08-30-22.03.00.868892	SYSDEFAULTSUBCLASS	0	0
2007-08-30-22.08.00.257679	ADMINS	21	110
2007-08-30-22.08.00.257679	BATCH	0	0
2007-08-30-22.08.00.257679	PROD_QRY	0	0
2007-08-30-22.08.00.257679	PROD_RPT	48	506010
2007-08-30-22.08.00.257679	SYSDEFAULTSUBCLASS	0	0

Figure 5-1 shows the total coordinator activity completed for each service subclass during the monitored time frame.

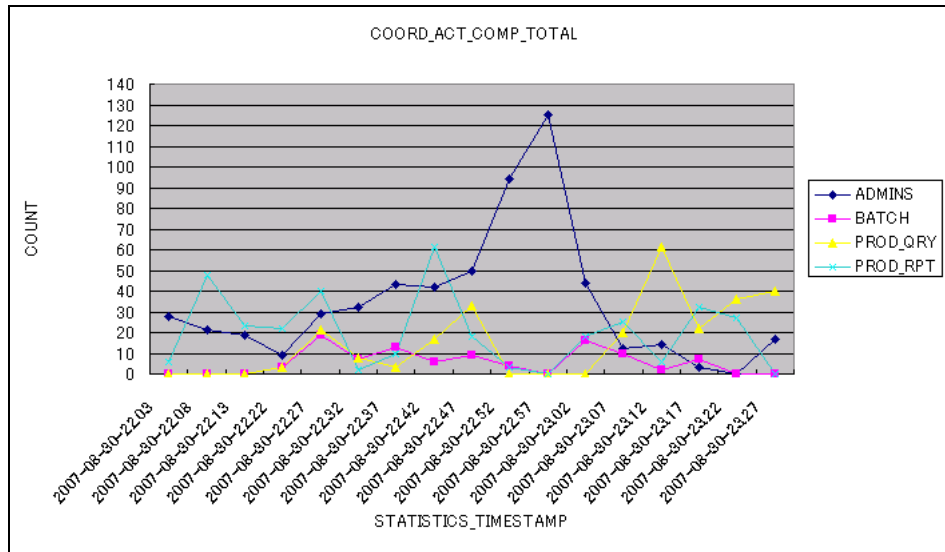


Figure 5-1 Total coordinator activity completed

Figure 5-2 shows the average of request execution time for each service subclass during the monitored time frame.

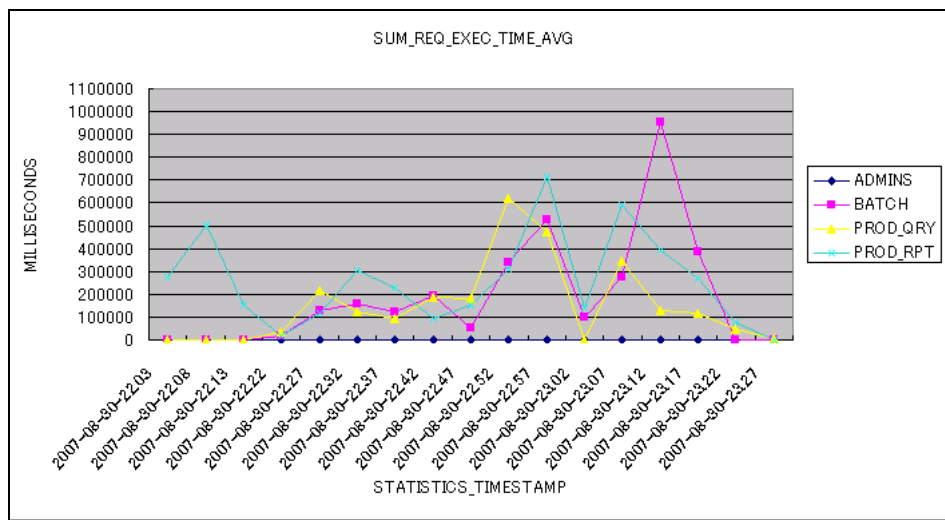


Figure 5-2 Request execution time average

From the graphs, we can easily see that at 2007-08-30-22:57, there was a big jump of total coordinator activity completed in ADMINS subclass. The request execution time average of BATCH, PROD_QRY and PROD_RPT also jumped

up at the same time. This means the ADMINS activities influenced the performance of BATCH, PROD_QRY and PROD_RPT. If you want to know what kind of activities ran at that time, you need to collect activity information by specifying COLLECT ACTIVITY DATA for the ADMINS service subclass. The data can help you decide whether to control the prefetch or CPU priority of ADMIN service subclass or not.

We also observed that at 2007-08-30-23.12, the average of request execution time of BATCH subclass jumped up. We found that was due to a BATCH user executed LOAD utility. The activity ran over 15 minutes. At that time the total coordinator activity completed for PROD_QRY subclass jumped up, but the request execution time average of PROD_QRY subclass declined slightly. This means the LOAD activity did not influenced the performance of PROD_QRY.

Analyzing the histogram

We want to know the tendency of the request execution time average on our system. This can be done by analyzing the ReqExecTime histogram.

Example 5-56 shows the view for the statistics event monitor table HISTOGRAMBIN_BASIC_MON. The view makes it easier to make a graph compared with service subclasses.

Example 5-56 View for the statistics event monitor table HISTOGRAMBIN_BASIC_MON

```
CREATE VIEW histograms_ret (
    histogram_type,
    service_superclass,
    service_subclass,
    bin_top,
    number_in_bin) AS

SELECT
    DISTINCT SUBSTR(histogram_type,1,24) AS histogram_type,
    SUBSTR(parentserviceclassname,1,24) AS service_superclass,
    SUBSTR(serviceclassname,1,24) AS service_subclass,
    top as bin_top,
    SUM(number_in_bin) AS number_in_bin

FROM
    histogrambin_basic_mon H,
    syscat.serviceclasses S

WHERE h.service_class_id=s.serviceclassid
AND histogram_type='ReqExecTime'
GROUP BY histogram_type, parentserviceclassname, serviceclassname, top;
--
--
CREATE VIEW hist_reqexectime (
    bin_top,
    admin_exec_time,
    batch_exec_time,
```

```

    prodrpt_exec_time,
    prodqry_exec_time) as
SELECT
DISTINCT x.bin_top,
(SELECT number_in_bin FROM histograms_ret a
 WHERE a.histogram_type='ReqExecTime' AND
       a.service_subclass='ADMINS' AND
       a.bin_top = x.bin_top) AS admin_exec_time,
(SELECT number_in_bin FROM histograms_ret b
 WHERE b.histogram_type='ReqExecTime' AND
       b.service_subclass='BATCH' AND
       b.bin_top = x.bin_top) AS batch_exec_time,
(SELECT number_in_bin FROM histograms_ret r
 WHERE r.histogram_type='ReqExecTime' AND
       r.service_subclass='PROD_RPT' AND
       r.bin_top = x.bin_top) AS prodrpt_exec_time,
(SELECT number_in_bin FROM histograms_ret q
 WHERE q.histogram_type='ReqExecTime' AND
       q.service_subclass='PROD_QRY' AND
       q.bin_top = x.bin_top) AS prodqry_exec_time
FROM histograms_ret x
;

```

Example 5-57 shows the ReqExecTime histogram for service subclasses.

Example 5-57 HIST_REQEXEETIME view sample data

> db2 "SELECT * FROM hist_reqexectime ORDER BY 1"

BIN_TOP	ADMIN_EXEC_TIME	BATCH_EXEC_TIME	PRODRPT_EXEC_TIME	PRODQRY_EXEC_TIME
-1	0	0	0	0
1	7769	1006	3246	1961
2	461	89	206	532
3	85	39	100	228
5	53	19	47	227
8	77	29	39	130
12	27	21	29	53
19	33	23	26	50
29	24	11	14	49
44	108	10	20	62
68	148	15	27	138
103	40	17	24	147
158	16	7	25	94
241	17	16	19	86
369	45	11	33	186
562	33	19	28	43
858	23	24	125	70
1309	0	50	127	78
1997	1	16	155	90
3046	3	38	133	101
4647	0	37	74	140
7089	0	38	175	64
10813	0	34	128	73
16493	0	40	117	101
25157	0	43	165	42

38373	0	74	257	57
58532	0	90	426	90
89280	0	107	362	86
136181	0	24	164	58
207720	0	37	87	86
316840	0	31	70	98
483283	0	38	40	47
737162	0	18	68	10
1124409	0	5	47	14
1715085	0	10	49	10
2616055	0	0	0	0
3990325	0	0	0	0
6086529	0	0	0	0
9283913	0	0	0	0
14160950	0	0	0	0
21600000	0	0	0	0

41 record(s) selected.

Figure 5-3 shows that the graph of the ReqExecTime histogram during the monitoring term. Most activities had a execution time between 0 and 1 milliseconds. Total 179 (98 + 47 + 10 + 14 + 10) activities in the PROD_QRY service subclass had a execution time between 316840 and 1715085 milliseconds (almost 5 to 28 minutes). As a result, we need to tune those queries to satisfy the business objectives.

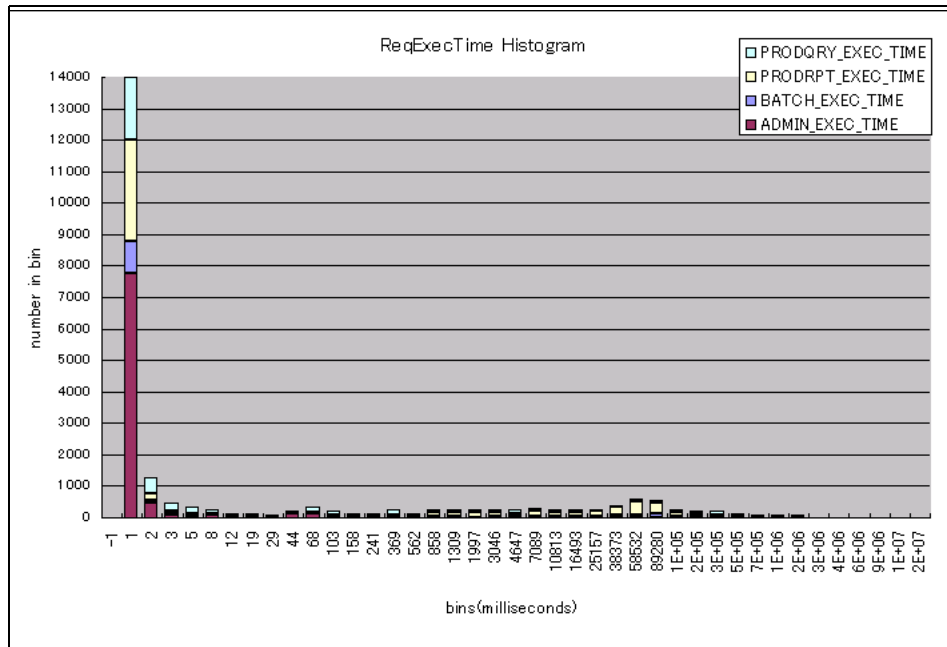


Figure 5-3 ReqExecTime histogram

5.3.2 Monitoring the queued job

In this section, we show you how to use table functions and event monitor to monitor the queued job. In this scenario, we allow the maximum of 50 database coordinator activities in the PROD_RPT service subclass. When the maximum number of database coordinator activities is exceeded, database coordinator activities are queued. Table 5-4 shows the modified PROD_RPT subclass configuration.

Table 5-4 Modified PROD_RPT service subclass configuration

Service subclass	Threshold	Threshold Action
PROD_RPT	QUEUE_THRESH: CONCURRENTDBCOORDA CTIVITIES > 50	Queuing CONTINUE

A threshold is added to the service subclass PROD_RPT to limit the number of concurrent coordinator activities to 50. See Example 5-58.

Example 5-58 limiting the number of activities

```
>db2 "CREATE THRESHOLD QUEUE_THRESH FOR SERVICE CLASS "PROD_RPT" UNDER
"HIGHLVL" ACTIVITIES ENFORCEMENT DATABASE WHEN CONCURRENTDBCOORDACTIVITIES > 50
CONTINUE";
DB20000I The SQL command completed successfully.
```

Queued workload occurrences

Example 5-59 shows the SQL statement for finding which workload occurrences are queued. We can see the application handle 4743, 4748, and 4750 are queued because those activities have QUEUED value in workload_occurrence_state.

Example 5-59 Which workload occurrences are queued

```
>db2 "SELECT substr(service_superclass_name,1,19) as superclass_name,
substr(service_subclass_name,1,19) as subclass_name, substr(workload_name,1,22)
as workload_name, substr(char(dbpartitionnum),1,4) as part, APPLICATION_HANDLE,
substr(application_name,1,10) as appl_name, WORKLOAD_OCCURRENCE_STATE FROM
TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES(' ', '-2'))"
```

SUPERCLASS_NAME	SUBCLASS_NAME	WORKLOAD_NAME	PART	APPLICATION_HANDLE
APPL_NAME	WORKLOAD_OCCURRENCE_STATE			

HIGHLVL	ADMINS	WL_ADMIN	0	4698
db2bp	UOWEXEC			

HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4722
dss.exe	UOWEXEC			
HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4723
dss.exe	UOWEXEC			
HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4728
dss.exe	UOWEXEC			
HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4743
dss.exe	QUEUED			
HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4744
dss.exe	UOWEXEC			
HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4745
dss.exe	UOWEXEC			
HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4746
dss.exe	UOWEXEC			
HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4747
dss.exe	UOWEXEC			
HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4748
dss.exe	QUEUED			
HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4749
dss.exe	UOWEXEC			
HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4750
dss.exe	QUEUED			
HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4751
dss.exe	UOWEXEC			
HIGHLVL	PROD_RPT	WL_PROD_RPT	0	4752
dss.exe	UOWEXEC			

14 record(s) selected.

Example 5-60 shows the SQL statement for listing the queued agents. In our example, three agents are queued.

Example 5-60 Agent queued

```
>db2 "SELECT application_handle, uow_id, activity_id,
substr(char(dbpartitionnum),1,4) as part, event_object, event_state FROM
TABLE(WLM_GET_SERVICE_CLASS_AGENTS('','',CAST(NULL AS BIGINT), -2)) WHERE
service_superclass_name <> 'SYSDEFAULTSYSTEMCLASS' event_object='WLM_QUEUE'
ORDER BY application_handle, uow_id, activity_id"
```

APPLICATION_HANDLE	UOW_ID	ACTIVITY_ID	PART	EVENT_OBJECT	EVENT_STATE
4743	10	1	0	WLM_QUEUE	IDLE
4748	9	1	0	WLM_QUEUE	IDLE
4750	7	1	0	WLM_QUEUE	IDLE

3 record(s) selected.

Example 5-61 shows how to find the activities that are queued but have not started executing yet. When an activity has entered the system but is in a queue, LOCAL_START_TIME field can be null.

Example 5-61 Activity in a queue but has not started executing

```
>db2 "SELECT application_handle, substr(char(dbpartitionnum),1,4) as part,
local_start_time, activity_state, activity_type FROM
TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(CAST(NULL AS BIGINT), -1))"
```

APPLICATION_HANDLE	PART	LOCAL_START_TIME	ACTIVITY_STATE	ACTIVITY_TYPE
4698	0	2007-08-30-21.13.11.933493	EXECUTING	READ_DML
4722	0	2007-08-30-21.10.40.257254	EXECUTING	READ_DML
4723	0	2007-08-30-21.09.08.795625	EXECUTING	READ_DML
4728	0	2007-08-30-21.10.35.907819	EXECUTING	READ_DML
4743	0	-	QUEUED	READ_DML
4744	0	2007-08-30-21.09.32.696410	EXECUTING	READ_DML
4745	0	2007-08-30-21.09.50.081916	EXECUTING	READ_DML
4746	0	2007-08-30-21.11.05.751756	EXECUTING	READ_DML
4747	0	2007-08-30-21.10.04.240474	EXECUTING	READ_DML
4748	0	-	QUEUED	READ_DML
4749	0	2007-08-30-21.10.52.481785	EXECUTING	READ_DML
4750	0	-	QUEUED	READ_DML
4751	0	2007-08-30-21.10.40.257259	EXECUTING	READ_DML
4752	0	2007-08-30-21.09.54.513369	EXECUTING	READ_DML

14 record(s) selected.

Example 5-62 shows the number of queued activities (current and total) and total time spent in the queue for the QUEUE_THRESH threshold. Five activities are in the queue and 14 activities were assigned to this queue since the last reset. 159091 milliseconds spent in the queue for 14 activities.

Example 5-62 Number of queued activities

```
>db2 "SELECT substr(threshold_name, 1, 15) threshname, threshold_predicate,
threshold_domain, dbpartitionnum part, queue_size_top, queue_size_current,
queue_time_total, queue_assignments_total queue_assign FROM
TABLE(WLM_GET_QUEUE_STATS('', '', '', -1))"
```

THRESHNAME	THRESHOLD_PREDICATE	THRESHOLD_DOMAIN	PART
QUEUE_SIZE_TOP	QUEUE_SIZE_CURRENT	QUEUE_TIME_TOTAL	QUEUE_ASSIGN
QUEUE_THRESH	CONCDBC	SB	0
5	5	159091	14

1 record(s) selected.

Threshold violations

Using the event monitor, you can see the activities that violate the threshold limit. Example 5-63 shows the SQL statement to query the event monitor table THRESHOLDVIOLATIONS_THRESH_MON and partial output. 49 activities are violated by the QUEUE_THRESH threshold.

Example 5-63 Querying event monitor table

```
>db2 "SELECT substr(char(agent_id),1,8) as agent_id,
substr(char(coord_partition_num),1,4) as part, threshold_action,
threshold_maxvalue, substr(threshold_predicate,1,28)
as threshold_predicate, time_of_violation FROM THRESHOLDVIOLATIONS_THRESH_MON"
```

AGENT_ID	PART	THRESHOLD_ACTION	THRESHOLD_MAXVALUE	THRESHOLD_PREDICATE
TIME_OF_VIOLATION				
-----	-----	-----	-----	-----
4215	0	Continue		5 ConcurrentDBCoordActivities
2007-08-30-16.10.11.000000				
4216	0	Continue		5 ConcurrentDBCoordActivities
2007-08-30-16.10.11.000000				
4209	0	Continue		5 ConcurrentDBCoordActivities
2007-08-30-16.10.12.000000				
4212	0	Continue		5 ConcurrentDBCoordActivities
...				
4743	0	Continue		10 ConcurrentDBCoordActivities
2007-08-30-21.10.40.000000				
4748	0	Continue		10 ConcurrentDBCoordActivities
2007-08-30-21.10.40.000000				
4746	0	Continue		10 ConcurrentDBCoordActivities
2007-08-30-21.10.52.000000				
4750	0	Continue		10 ConcurrentDBCoordActivities
2007-08-30-21.10.52.000000				
4743	0	Continue		10 ConcurrentDBCoordActivities
2007-08-30-21.10.59.000000				
4748	0	Continue		10 ConcurrentDBCoordActivities
2007-08-30-21.11.05.000000				

49 record(s) selected.

Activity queue time

Histogram is helpful in finding the activity queue time in a time frame. Example 5-64 shows the CoordActQueueTime histogram for the PROD_RPT service subclass. In the output, we can see that 342 activities spent for between 0 and 1 milliseconds in threshold queues before it starts executing. Although 3 activities spent for over 3 minutes in threshold queues, we concluded that it was a permissible range.

Example 5-64 CoordActQueueTime histogram

```
>db2 "create view histograms (histogram_type, service_superclass,
service_subclass, bin_top, number_in_bin) as select distinct
```

```

substr(histogram_type,1,24) as histogram_type,
substr(parentserviceclassname,1,24) as service_superclass,
substr(serviceclassname,1,24) as service_subclass, top as bin_top,
sum(number_in_bin) as number_in_bin from histogrambin_db2statistics h,
syscat.serviceclasses s where h.service_class_id = s.serviceclassid group by
histogram_type, parentserviceclassname, serviceclassname, top"
DB20000I The SQL command completed successfully.

```

```

>db2 "SELECT bin_top, number_in_bin FROM histograms WHERE
histogram_type='CoordActQueueTime' and service_subclass='PROD_RPT' ORDER BY
bin_top"

```

BIN_TOP	NUMBER_IN_BIN
-1	0
1	342
2	0
3	0
5	0
8	0
12	0
19	0
29	0
44	0
68	0
103	0
158	0
241	0
369	0
562	1
858	1
1309	0
1997	2
3046	3
4647	0
7089	2
10813	2
16493	3
25157	7
38373	2
58532	9
89280	1
136181	0
207720	3
316840	0
483283	0
737162	0
1124409	0
1715085	0

2616055	0
3990325	0
6086529	0
9283913	0
14160950	0
21600000	0

41 record(s) selected.

5.3.3 Identifying query with long runtime

A common business requirement for ad hoc queries is that the database system must complete a percentage of queries for under a time frame, for example, complete 90% of queries for under 5 minutes. To meet this business objective, first task is to identify those queries with long runtime. You can then use Design Advisor to tune these queries.

In this scenario, we identify the long run queries under the PROD_QRY service subclass. We define a threshold for PROD_QRY to identify queries run over 5 minutes. The threshold action is CONTINUE and the COLLECT ACTIVITY DATA WITH DETAILES option is specified so that an activity that violates the threshold is sent to the activities event monitor on completion.

Table 5-5 shows the updated workload management configuration.

Table 5-5 Updated PROD_QRY service subclass configuration

Service subclass	Threshold	Threshold Action
PROD_QRY	LONGRUN: ACTIVITYTOTALTIME > 5 minutes	COLLECT ACTIVITY DATA WITH DETAILS CONTINUE

We use following steps to identify queries with long runtime:

1. Create and activate a threshold violations event monitor and an activity event monitor.

Example 5-65 shows how to create and activate event monitor tables.

Example 5-65 Creating and activating event monitor tables

```
CREATE EVENT MONITOR act_mon FOR ACTIVITIES WRITE TO TABLE AUTOSTART
DB20000I The SQL command completed successfully.
```

```
CREATE EVENT MONITOR thresh_mon FOR THRESHOLD VIOLATIONS WRITE TO TABLE
AUTOSTART
DB20000I The SQL command completed successfully.
```

```
SET EVENT MONITOR act_mon STATE 1
DB20000I The SQL command completed successfully.
```

```
SET EVENT MONITOR thresh_mon STATE 1
DB20000I The SQL command completed successfully.
```

To import activity information into the Design Advisor, the activity event monitor table is needed, not file. You can see how many activities were violated in the threshold violations event monitor. You can investigate what query were violated in the activities event monitor.

We create six event monitor tables:

- For activity event monitor ACT_MON
 - ACTIVITYSTMT_ACT_MON
 - ACTIVITYVALS_ACT_MON
 - ACTIVITY_ACT_MON
 - CONTROL_ACT_MON
- For threshold violations event monitor THRESH_MON
 - CONTROL_THRESH_MON
 - THRESHOLDVIOLATIONS_THRESH_MON

2. Use the CREATE THRESHOLD statement to collect activity information. Example 5-66 shows how to create a threshold to collect activity information.

Example 5-66 Creating threshold

```
>db2 "create threshold LONGRUN for service class PROD_QRY under HIGHLVL
activities enforcement database when activitytotaltime > 5 minutes collect
activity data with details continue"
DB20000I The SQL command completed successfully.
```

```
>db2 "SELECT substr(thresholdname,1,12) as thresh_name, thresholdid FROM
syscat.thresholds"
```

THRESH_NAME	THRESHOLDID
LONGRUN	3
QUEUE_THRESH	2

2 record(s) selected.

3. Identify the statement text and the total threshold violations.

You can see the statement text by selecting the activities event monitor table. Example 5-67 shows the queries run over 5 minutes.

Example 5-67 Identifying queries run over 5 minutes

```
>db2 "SELECT substr(stmt_text,1,200) as stmt_text FROM
activitystmt_act_mon"
```

```
<<extract of output>>
```

```
STMT_TEXT
```

```
-----
SELECT COL1, COL2, COLBB FROM T1, T2 WHERE COL1 = COLAA
SELECT COL1, COL2, COLBB FROM T1, T2 WHERE COL1 = COLAA
SELECT COL1, COL2, COLBB FROM T1, T2 WHERE COL1 = COLAA
```

You can see the total threshold violation count by selecting threshold violations event monitor table. Example 5-68 shows the number of threshold violation occurred.

Example 5-68 Number of threshold violations occurred

```
db2 "SELECT count(*) as thresh_violation_cnt FROM
thresholdviolations_thresh_mon WHERE thresholdid=3"
```

```
THRESH_VIOLATION_CNT
```

```
-----
                        8
```

```
1 record(s) selected.
```

4. Use the db2advvis command to import activity information into the Design Advisor.

```
>db2advvis -d WLMDB -wlm ACT_MON serviceclass HIGHLVL,PROD_QRY
```

We specify the service superclass and subclass name in this example.



WLM Sample Scenario - OLTP

OLTP systems are highly process oriented and contains present data. Most of the OLTP systems requires 24x7 availability and has the primary business objective as maintaining a quick response time.

In this chapter, we demonstrate chapter how to how to use DB2 WLM to achieve the primary business objective for an OLTP system. We cover briefly how to convert the management requirements to workloads and implement in DB2 WLM. We show how to monitor and analyze the data to verify if the business requirements are achieved.

6.1 Business objectives

Maintaining a consistent transaction response time is typically the primary and often is the only objective of an OLTP system. In the real-time environment, achieving this goal can be a challenging task as many applications from different business units simultaneously run on the system and using the same database.

In this example OLTP system, three business units, Sales, Accounting, and Inventory are sharing the resources and have the following characteristics:

- ▶ The database is shared by multiple business groups. All the business units have equal share of resources.
- ▶ ETL runs off hours and doesn't interfere with business user workload.
- ▶ Administrative tasks and scheduled maintenance tasks runs off hours or under scheduled maintenance period and don't interfere with business user workload.

The business objectives are stated as follows:

- ▶ Maintain sub-second response times for all queries from Sales team.
- ▶ Prevent excessive concurrency of large queries hog the system.
- ▶ Mitigate long running queries from all departments.
- ▶ Prevent long idle connections.

We demonstrate how to use the WLM to manage and control the workload to achieve the response time requirement.

6.2 Identification

From the business objectives, we describe the management requirements as follows:

- ▶ Sales applications require a consistent sub-second response times for all queries.
- ▶ The majority queries from Inventory and Accounting departments have no response time requirements.
- ▶ Mitigate runaway queries with long execution times.
- ▶ To prevent large number of concurrent queries affecting the overall performance of the system, limited the concurrent activities to 10.
- ▶ The maximum allowed time for a database connection to be idle is 30 minutes.

We broadly categorize database activities that we used to identify the workload to address the management requirements as follows:

- ▶ Based on time duration:
Database activities that are time based. For example, each OLTP request has to be completed in less than a second.
- ▶ Based on limits:
Activities that can be categorized based on limits. For example, limits on resource or cardinality.
- ▶ Based on action:
What we need to take action if the goal were not met. For example, to continue or stop.

We also gather additional details needed to formulate WLM solutions. For example, what are the sources of work? where do they come from, user, application, or tools? There are many different criteria that could be used to identify database activities. The following are the additional sources of work information available and categorized:

Table 6-1 lists the user IDs, groups, and applications on our sample system.

Table 6-1 User categories and applications

User	Group	Tools	Application name
salesusr	salesgrp		sales.exe
invusr	invgrp		inv.exe
accusr	accgrp		acct.exe
salesmgr	salesgrp	cognos	sales.exe, salesrpt.exe
invmgr	invgrp		inv.exe, salesrpt.exe
accmgr	accgrp		acct.exe, salesrpt.exe

6.3 Consistent response time

To achieve the management requirements of having a consistent sub-second response times for all queries from Sales applications, we want to give the Sales applications higher priority and operatively report the users, applications, and queries that do not meet the performance requirements so we can address the problem.

Table 6-2 shows the worksheet that summarize the workload for this example.

Table 6-2 Worksheet - consistent response time

Task	Business requirements	Identification	Action
Sales	Maintain sub-second response time	Client user ID = SALESUSR Client application name = sales.exe or oltp.exe	<ul style="list-style-type: none"> ▶ Give higher priority to Sales applicaitons. ▶ Report activity details such as the user ID, appliation name, query that do not meet the business requirement

6.3.1 Define DB2 workloads and service classes

We use the worksheet to define workloads and service classes. In this example, we represent each business unit as a single workload but focus on achieving the business objective for Sales department. Sales workload can be identified with user IDs of the Sales business group or application names.

As for the service classes, since all the business units share the same system, we define one super service class HIGHLVL and a subclass for each business unit under that superclass. If a workload is created without specifying any service class, by default, it will be associated with SYSDEFAULTUSERCLASS. We have all our unclassified work run under the default service class.

Table 6-3 shows the association between workload and service classes.

Table 6-3 Workloads and service classes

Work category	DB2 workload	DB2 service classes
Sales	WL_SALES	SC_SALES under HIGHLVL
Accounting	WL_ACCT	SC_ACC under HIGHLVL
Inventory	WL_INVENT	SC_INV under HIGHLVL
All other work	Default workload	Default service class

Define controls

To maintain the sales applications in sub-section response time, we have two actions in place:

- ▶ Giving higher priority

Prefetchers retrieve data from disk, and storing this data in buffer pools so that it can be quickly accessed by agents. DB2 WLM provides you the capability to set prefetch priority or agent priority for application. Since the

Sales application uses prefetching, we use the service class prefetch priority to give the sales related work higher priority. You can specify the prefetch priority of a service class with the PREFETCH PRIORITY option on either the CREATE or ALTER SERVICE CLASS statement.

► Report activity details

To report the information about activity we need to specify COLLECT ACTIVITY DATA, COLLECT AGGREGATE ACTIVITY, and REQUEST DATA so that the information about each activity that is executed in this service class is to be sent to the applicable event monitor when the activity completes.

WLM object definitions

Example 6-1 shows the WLM object definitions.

Example 6-1 Service class, workload definition for Goal1

```
CONNECT TO WLMDB;
-- for OLTP workloads
SET WORKLOAD TO SYSDEFAULTADMWORKLOAD;
-----
Create service classes
-----
CREATE SERVICE CLASS highlvl DISABLE;
CREATE SERVICE CLASS sc_sales UNDER highlvl AGENT PRIORITY -10 PREFETCH
PRIORITY high COLLECT ACTIVITY DATA ON ALL WITH DETAILS COLLECT AGGREGATE
ACTIVITY DATA EXTENDED COLLECT AGGREGATE REQUEST DATA DISABLE;

CREATE SERVICE CLASS sc_inv UNDER highlvl COLLECT ACTIVITY DATA ON ALL WITH
DETAILS COLLECT AGGREGATE ACTIVITY DATA EXTENDED COLLECT AGGREGATE REQUEST DATA
DISABLE;

CREATE SERVICE CLASS sc_acc UNDER highlvl COLLECT ACTIVITY DATA ON ALL WITH
DETAILS COLLECT AGGREGATE ACTIVITY DATA EXTENDED COLLECT AGGREGATE REQUEST DATA
DISABLE;
-----
-- Create workloads that map the connections to the service classes
-----
CREATE WORKLOAD w1_sales CURRENT CLIENT_USERID ('SALESUSR') CURRENT
CLIENT_APPLNAME ('sales.exe','oltp.exe') DISABLE SERVICE CLASS sc_sales UNDER
highlvl POSITION AT 1 COLLECT ACTIVITY DATA WITH DETAILS;

CREATE WORKLOAD w1_invent CURRENT CLIENT_USERID ('INVUSR') CURRENT
CLIENT_APPLNAME ('inv.exe') DISABLE SERVICE CLASS SC_INV UNDER highlvl POSITION
AT 2;
```

```
CREATE WORKLOAD w1_acct CURRENT CLIENT_USERID ('ACCUSR') CURRENT
CLIENT_APPLNAME ('acct.exe') DISABLE SERVICE CLASS SC_ACC UNDER highlvl
POSITION AT 3;
```

```
-----
-- Grant usage on the workloads to PUBLIC
-----
```

```
GRANT USAGE ON WORKLOAD w1_sales TO PUBLIC;
GRANT USAGE ON WORKLOAD w1_invent TO PUBLIC;
GRANT USAGE ON WORKLOAD w1_acct TO PUBLIC;
```

```
-----
-- Enable Service classes and Workloads
-----
```

```
ALTER SERVICE CLASS highlvl ENABLE;
ALTER SERVICE CLASS sc_sales UNDER highlvl ENABLE;
ALTER SERVICE CLASS sc_inv UNDER highlvl ENABLE;
ALTER SERVICE CLASS sc_acc UNDER highlvl ENABLE;
ALTER WORKLOAD w1_sales ENABLE;
ALTER WORKLOAD w1_invent ENABLE;
ALTER WORKLOAD w1_acct ENABLE;
```

If DEFAULT is specified for a service superclass, this equates to a prefetch priority of “medium” for the service superclass. You can specify a different prefetch priority for any service subclass in the service superclass, but if you use the default prefetch priority for the service subclass, the service subclass inherits its prefetch priority setting from the service superclass.

For our example we set the prefetch priority to HIGH at subclass level, so that sales subclass get the high prefetch priority over others.

Note: If you are using the DB2_HI_PRI_PREFETCH_AUTHID or DB2_LO_PRI_PREFETCH_AUTHID environment variables to route prefetch requests to a specific prefetch queue based on an authorization ID, you should use service class prefetch priorities to ensure that your workload continues to be processed as expected.

If you have AIX WLM running, you can map each DB2 service class to a corresponding AIX WLM service class to provide broader control of the workloads.

6.3.2 Monitoring

Once the service classes and workloads defined, we monitor the system to see if our business objective is met and the controls we placed work as expected:

- ▶ Check workloads, service class, and other WLM components.

- ▶ Collect information required for analysis
 - Collect the response time for all requests coming from Sales representative and Sales application.

Make sure that WLM_COLLECT_INT parameter is set to a value more than zero. For our monitoring, we set this value to 5.

The monitoring tasks we performed include the following:

- ▶ Check the Service classes, workloads, and their evaluation order.

Example 6-2 shows the SQL statements and output. Since all the subclasses are under one superclass, we want to make sure that the SC_SALES will be evaluated first.

Example 6-2 Checking service classes and workloads definition and evaluation order

```
db2 "select SUBSTR(SERVICECLASSNAME,1,27) as SC,
SUBSTR(PARENTSERVICECLASSNAME,1,27) as PARENTCLASS, ENABLED from
syscat.serviceclasses"
```

SC	PARENTCLASS	ENABLED
SYSDEFAULTSUBCLASS	SYSDEFAULTSYSTEMCLASS	Y
SYSDEFAULTSUBCLASS	SYSDEFAULTMAINTENANCECLASS	Y
SYSDEFAULTSUBCLASS	SYSDEFAULTUSERCLASS	Y
SYSDEFAULTSUBCLASS	HIGHLVL	Y
SC_SALES	HIGHLVL	Y
SC_ACC	HIGHLVL	Y
SC_INV	HIGHLVL	Y
SYSDEFAULTSYSTEMCLASS	-	Y
SYSDEFAULTMAINTENANCECLASS	-	Y
SYSDEFAULTUSERCLASS	-	Y
HIGHLVL	-	Y

11 record(s) selected.

```
->db2 "select WORKLOADID as ID, EVALUATIONORDER as EORDER,
substr(WORKLOADNAME,1,25) as WORKLOAD, ENABLED,
substr(SERVICECLASSNAME,1,25) as SCNAME from syscat.workloads order by 2"
```

ID	EORDER	WORKLOAD	ENABLED	SCNAME
5	1	WL_SALES	Y	SC_SALES
3	2	WL_INVENT	Y	SC_INV
4	3	WL_ACCT	Y	SC_ACC
1	4	SYSDEFAULTUSERWORKLOAD	Y	SYSDEFAULTSUBCLASS
2	5	SYSDEFAULTADMWORKLOAD	Y	SYSDEFAULTSUBCLASS

5 record(s) selected.

- ▶ Determine to which workload the applications are assigned.

We use WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURANCES table function to find to which workload the Sales applications are assigned,. Refer to Chapter 5, “Monitoring” on page 83 for workload manager table functions.

Example 6-3 shows that one of the Sales application oltp.exe is assigned to SC_SALES service class and WL_SALES workload.

Example 6-3 Workloads and service class

```
->db2 "select substr(application_name,1,15) as application_name ,
substr(CLIENT_USER,1,10) as client_id , substr(CLIENT_APPLNAME,1,15) as
client_applname , substr(service_subclass_name,1,15) as sub_class ,
substr(workload_name,1,20) as workload_name ,
substr(workload_occurrence_state,1,10) as wlm_state from
table(wlm_get_service_class_workload_occurrences('','",-1))"
```

APPLICATION_NAME	CLIENT_ID	CLIENT_APPLNAME	SUB_CLASS	WORKLOAD_NAME	WLM_STATE
db2bp	-	-	ADMINS	WL_ADMIN	UOWWAIT
db2bp	-	-	ADMINS	WL_ADMIN	UOWEXEC
oltp.exe	-	-	SC_SALES	WL_SALES	UOWEXEC
oltp.exe	-	-	SC_SALES	WL_SALES	UOWEXEC
oltp.exe	-	-	SC_SALES	WL_SALES	UOWEXEC
oltp.exe	-	-	SC_SALES	WL_SALES	UOWEXEC
oltp.exe	-	-	SC_SALES	WL_SALES	UOWEXEC
oltp.exe	-	-	SC_SALES	WL_SALES	UOWEXEC
oltp.exe	-	-	SC_SALES	WL_SALES	UOWEXEC
oltp.exe	-	-	SC_SALES	WL_SALES	UOWEXEC
oltp.exe	-	-	SC_SALES	WL_SALES	UOWEXEC
...					

- Check agent priority and prefetch priority

We can check the agent and prefetch priority for sales.exe or oltp.exe using SNAP_GET_APPL_V95 table function or db2pd utility. db2pd is used here to display both the agent and prefetch priority for SC_SALES service class:

```
db2pd -db w1mdb -serviceclasses
```

Example 6-4 shows agent and prefetch priority values.

Example 6-4 Agent and Prefetch priority in Service classes

```
->db2pd -db w1mdb -serviceclasses
...
Service Class Name      = SC_SALES
Service Class ID       = 21
Service Class Type     = Service Subclass
Parent Superclass ID   = 14
Service Class State    = Enabled
Agent Priority         = -10
Prefetch Priority      = High
Outbound Correlator    = None
Collect Activity Opt   = On all partitions with details
Collect Aggr Activity Opt = Extended
Collect Aggr Request Opt = Base
Act Lifetime Histogram Template ID = 1
Act Queue Time Histogram Template ID = 1
Act Execute Time Histogram Template ID = 1
Act Estimated Cost Histogram Template ID = 1
Act Interarrival Time Histogram Template ID = 1
Request Execute Time Histogram Template ID = 1
```

```

Access Count          = 7
Last Stats Reset Time = 09/19/2007 14:58:40.000000
Activities HWM        = 8
Activities Completed  = 41
Activities Rejected   = 0
Activities Aborted    = 0

```

```

Associated Agents:
EDU ID      AppHandl [nod-index]  WL ID      WLO ID      UOW ID      Activity ID
50135153246212  28717  [000-28717]  11          76          6          1
43516608643076  28718  [000-28718]  11          77          5          1
42412802048004  28719  [000-28719]  11          78          6          1
71133248356356  28720  [000-28720]  11          79          5          1
41308995452932  28722  [000-28722]  11          81          5          1
46823733460996  28724  [000-28724]  11          82          5          1
49031346651140  28723  [000-28723]  11          83          5          1
.....

```

► Response time for Sales application

To get the response time, we used the histogram template and collected request execution time (ReqExecTime) for Sales service class. We created an histogram view to collect required information as shown in Example 6-5.

Example 6-5 Create Histogram view

```

CREATE VIEW HISTOGRAMS (histogram_type,
                        service_superclass,
                        service_subclass,
                        bin_top,
                        number_in_bin) AS
SELECT
DISTINCT SUBSTR(histogram_type,1,24) AS histogram_type,
SUBSTR(parentserviceclassname,1,24) AS service_superclass,
SUBSTR(serviceclassname,1,24) AS service_subclass,
TOP AS bin_top,
SUM(number_in_bin) AS number_in_bin
FROM
    histogrambin_basic_mon H,
    syscat.serviceclasses S
WHERE h.service_class_id=s.serviceclassid
GROUP BY histogram_type, parentserviceclassname, serviceclassname, top;

```

Once the histogram view is created, select the output from the view to see the request execution time output for Sales service class. See Example 6-6.

Example 6-6 Histogram output for SALES service class

```

->db2 "select BIN_TOP, substr(SERVICE_SUBCLASS,1,15) as SC_CLASS,
NUMBER_IN_BIN FROM histograms where HISTOGRAM_TYPE = 'ReqExecTime' and
SERVICE_SUBCLASS = 'SC_SALES' order by 1,2"

```

```

BIN_TOP      SC_CLASS      NUMBER_IN_BIN
-----
-1 SC_SALES      0

```

1	SC_SALES	1503
2	SC_SALES	12
3	SC_SALES	0
5	SC_SALES	0
8	SC_SALES	0
12	SC_SALES	0
19	SC_SALES	0
29	SC_SALES	0
44	SC_SALES	0
68	SC_SALES	0
103	SC_SALES	0
158	SC_SALES	0
241	SC_SALES	0
369	SC_SALES	18
562	SC_SALES	8
858	SC_SALES	0
1309	SC_SALES	0
1997	SC_SALES	0
3046	SC_SALES	20
4647	SC_SALES	0
7089	SC_SALES	0
10813	SC_SALES	0
16493	SC_SALES	0
25157	SC_SALES	0
38373	SC_SALES	0
58532	SC_SALES	0
89280	SC_SALES	0
136181	SC_SALES	0
207720	SC_SALES	0
316840	SC_SALES	0
483283	SC_SALES	0
737162	SC_SALES	0
1124409	SC_SALES	0
1715085	SC_SALES	0
2616055	SC_SALES	0
3990325	SC_SALES	0
6086529	SC_SALES	0
9283913	SC_SALES	0
14160950	SC_SALES	0
21600000	SC_SALES	0

41 record(s) selected.

Figure 6-1 shows the Sales application response time chart.

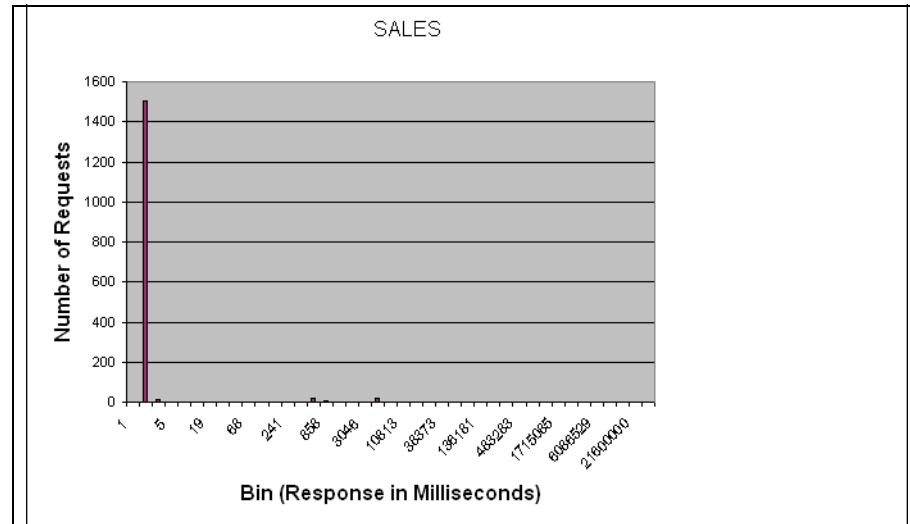


Figure 6-1 Histogram showing Sales queries response times

Analysis

After collecting the information those are required to check and validate our goal, now we can do some analysis on the information we collected. This will be used to validate our WLM definitions and to see whether we are near our goals.

From the Histogram view, we can see the response time for request execution time in milliseconds. We can see most of the responses fall under 1 sec. In our output taken for a short period of one hour, 58 out of 1561 requests response time falls outside our Goal, that is around 3.7%. Our objective is to have all the 100% requests from sales have <1 sec. response time. Objective is partially met. Further analysis on those 58 requests has to be performed and validate the SQL statements.

Collecting long running queries

To collect additional information about the queries running more than a second, we need to collect the information from activity monitor data. The creation of activity monitors are shown in Example 6-7

Example 6-7 Create event monitor s

```
CREATE EVENT MONITOR basic_mon FOR STATISTICS WRITE TO TABLE
control (TABLE control_basic_mon, IN maint ),
histogrambin (TABLE histogrambin_basic_mon, IN maint ),
qstats (TABLE qstats_basic_mon, IN maint ),
scstats (TABLE scstats_basic_mon, IN maint ),
wcstats (TABLE wcstats_basic_mon, IN maint ),
```

```
wlstats (TABLE wlstats_basic_mon, IN maint ) AUTOSTART;

CREATE EVENT MONITOR act_mon FOR ACTIVITIES WRITE TO TABLE
  activity (TABLE activity_act_mon, IN maint ),
  activitystmt (TABLE activitystmt_act_mon, IN maint ),
  activityvals (TABLE activityvals_act_mon, IN maint ),
  control (TABLE control_act_mon, IN maint ) AUTOSTART;
```

Next, create a view from activity monitor table, to collect the query information which is running morethan a second. Using TIME_COMPLETED and TIME_STARTED from activity table, we calculated the time taken for activity to complete.

Example 6-8 shows the create view statement to collect the activites for sales service subclass and from applications 'oltp.exe' and 'sales.exe'

Example 6-8 Create view

```
CREATE VIEW SALES_RESP (APPLID, AGENTID, APPL_NAME, SUBCLASS, SUPERCLASS,
WORKLOADID, TOTALTIME) AS SELECT A.APPL_ID, A.AGENT_ID, A.APPL_NAME,
A.SERVICE_SUBCLASS_NAME, A.SERVICE_SUPERCLASS_NAME, WORKLOAD_ID, (time_completed
- time_started) from ACTIVITY_ACT_MON A, ACTIVITYSTMT_ACT_MON B WHERE
(B.APPL_ID = A.APPL_ID) and (A.APPL_NAME = 'oltp.exe' OR
A.APPL_NAME = 'sales.exe') and (A.SERVICE_SUBCLASS_NAME = 'SC_SALES') and
((A.TIME_COMPLETED - A.TIME_STARTED) > 1))
```

Using the SALES_RESP view, we can see the application id, application name for queries running morethan one second. Example 6-9. shows the output

Example 6-9 Output from Sales_resp view

```
-->db2 "select substr(applid,1,30) as applid, substr(appl_name,1,30) as
appln_name, totaltime from sales_resp"
```

APPLID	APPLN_NAME	TOTALTIME
*N3.db2inst1.070913022101	oltp.exe	2.202352
*N3.db2inst1.070913022101	oltp.exe	1.718462
*N3.db2inst1.070913022101	oltp.exe	1.601447
*N3.db2inst1.070913022101	oltp.exe	2.385395
*N3.db2inst1.070913022009	oltp.exe	1.301762
*N3.db2inst1.070913022009	oltp.exe	1.105287
*N3.db2inst1.070913022009	oltp.exe	1.357266
*N3.db2inst1.070913022009	oltp.exe	1.098919
*N3.db2inst1.070913022007	oltp.exe	1.248636
*N3.db2inst1.070913022007	oltp.exe	1.421740
*N3.db2inst1.070913022012	oltp.exe	1.350745
*N3.db2inst1.070913022012	oltp.exe	1.357736

6.4.1 Define DB2 workloads and service classes

Additional workload and service class definitions are not required. The workload and service class definitions defined in 6.3, “Consistent response time” on page 153 are sufficient.

6.4.2 Define controls

You can create an activity threshold of type ACTIVITYTOTALTIME and perform this database wide. In this example, we create a threshold and specify that when the limit is reached, continue query execution and collect data for later analysis. Table 6-4 shows the activity name and its description.

Table 6-4 Threshold definition

Threshold	Description
TH_DB_TATIME	Threshold for database wide to check for total activity time for more than 30 minutes. Action: Continue and collect activity data with details.

COLLECT ACTIVITY DATA clause of the CREATE THRESHOLD statement specifies that for each activity that exceeded the threshold, send the data the active event monitor when the activity completes.

Example 6-11 shows a new threshold TH_DB_TATIME is created to check total activity time of any database activity.

Example 6-11 Create threshold to check total activity time with database scope

```
CREATE THRESHOLD th_db_tatime FOR DATABASE ACTIVITIES ENFORCEMENT DATABASE
DISABLE WHEN ACTIVITYTOTALTIME > 30 MINUTES COLLECT ACTIVITY DATA WITH DETAILS
CONTINUE;
ALTER THRESHOLD th_db_tatime ENABLE;
```

Ensure that required event monitors are active to monitor the events.

Example 6-7 on page 161 shows the creation of event monitors. Example 6-12 shows the current active event monitors.

Example 6-12 Active event monitors

```
->db2 "SELECT SUBSTR(EVMONNAME,1,20)AS EVMONNAME, CASE WHEN
EVENT_MON_STATE(EVMONNAME) = 0 THEN 'Inactive' WHEN EVENT_MON_STATE(EVMONNAME)
= 1 THEN 'Active' END AS EVSTATE FROM SYSCAT.EVENTMONITORS ORDER BY 2"
```

```

EVMONNAME          EVSTATE
-----
ACT_MON            Active
```

```

BASIC_MON           Active
DB2DETAILDEADLOCK  Active
THRESH_MON         Active
BASIC_MON2        Inactive
DB2STATISTICS     Inactive
WLM_THRESH        Inactive

```

7 record(s) selected.

Example 6-13 show the threshold we created till now and display its domain and enforcements.

Example 6-13 List thresholds

```
->db2 "SELECT SUBSTR(THRESHOLDNAME,1,15) AS thresname,origin, enabled,
thresholdid, enforcement, queuesize, collectactdata FROM SYSCAT.THRESHOLDS"
```

```

THRESNAME          ORIGIN  ENABLED  THRESHOLDID  ENFORCEMENT  QUEUESIZE  COLLECTACTDATA
-----
TH_DB_TATIME       U       Y              1 D              0 D

```

1 record(s) selected.

6.4.3 Monitoring

Example 6-14 shows the activities that violated threshold by the activity total time of more than 30 minutes. The queries are logged and allowed to continue.

Example 6-14 Threshold violation showing activity total time exceeded

```
->db2 "SELECT appl_id, threshold_action, threshold_maxvalue,
SUBSTR(threshold_predicate,1,28) AS threshold_predicate, time_of_violation from
thresholdviolations_thresh_mon ORDER BY 4"
```

```

APPL_ID              THRESHOLD_ACTION  THRESHOLD_MAXVALUE  THRESHOLD_PREDICATE
TIME_OF_VIOLATION
-----
*NO.db2inst1.070913062239  Continue          1800 ActivityTotalTime
2007-09-13-02.03.14.000000
*NO.db2inst1.070913062840  Continue          1800 ActivityTotalTime
2007-09-13-02.03.14.000000
*NO.db2inst1.070913062839  Continue          1800 ActivityTotalTime
2007-09-13-02.03.14.000000
*NO.db2inst1.070913062305  Continue          1800 ActivityTotalTime
2007-09-13-02.03.14.000000
*NO.db2inst1.070913062309  Continue          1800 ActivityTotalTime
2007-09-13-02.03.15.000000
*NO.db2inst1.070913062238  Continue          1800 ActivityTotalTime
2007-09-13-02.03.15.000000
*NO.db2inst1.070913062306  Continue          1800 ActivityTotalTime
2007-09-13-02.03.15.000000
...

```

With the application ID, we can collect the SQL statement text and find out who or which application the long running query is coming from.

Example 6-15 shows the SQL statement text obtained from activity statement monitor ACTIVITYSTMT_ACT_MON.

Example 6-15 Get Threshold violation statement text

```
->db2 "select appl_id, substr(STMT_TEXT,1,400) as STMT_TEXT from
ACTIVITYSTMT_ACT_MON where appl_id = '*NO.db2inst1.070913062239'"
```

```
APPL_ID
-----
STMT_TEXT
-----
*NO.db2inst1.070913062239
select o_orderpriority, count(*) as order_count from tpcc.orders where o_orderdate >= date
('1996-04-01') and o_orderdate < date ('1996-04-01') + 3 month and exists ( select * from
tpcc.lineitem where l_orderkey = o_orderkey and l_commitdate < l_receiptdate ) group by
o_orderpriority order by o_orderpriority

1 record(s) selected.
```

In Example 6-16, we find that the problem query is from INVMGR user running inv.exe application. This information is collected from ACTIVITY_ACT_MON table.

Example 6-16 Identifying the application information

```
->db2 "SELECT SUBSTR(appl_name,1,20) AS applname,
SUBSTR(service_subclass_name,1,20) AS subclassname,
SUBSTR(service_superclass_name,1,20) AS superclass, session_auth_id FROM
activity_act_mon WHERE appl_id = '*NO.db2inst1.070913062239'"
```

```
APPLNAME  SUBCLASSNAME  SUPERCLASS  SESSION_AUTH_ID
-----
inv.exe   SC_INV        HIGHLVL    INVMGR

1 record(s) selected.
```

6.4.4 Summary

From the analysis of monitoring data, we know that our objective has been met partially. We also found that the long running applications are from Inventory department. We now need to develop a management strategy to limit the number of expensive queries. We can create a separate service class and assign a lower priority for the inventory reports.

6.5 Prevent concurrent queries hogging the system

In this section, we address how to prevent too many large queries running concurrently that is hogging the system and impacts the performance of other applications.

Additional information can be collected for future use such as capacity planning.

In this example, the management requirement is to queue the READ queries with high costs (`TIMERONCOST > 100000.0`) if the concurrent queries of this type is more than 10 running against the database. This is for demonstrating purpose. Usually, it will be more for an OLTP system.

6.5.1 Identification

We use the workload and service classes defined in Appendix 6.3, “Consistent response time” on page 153 for this management requirement. For this example, we use WLM objects work class and work type.

6.5.2 Define work classes for control

A work action effectively provides an action that can be applied to a set of work classes. A work action set can contain one or more work actions that can be applied to a specific superclass or to the database as a whole.

If the work action set is defined for a service superclass, then typically the work action set would simply map the activity to a service subclass and have thresholds defined on the subclass aiding the management of the activity. The work action set also can be defined for the database if one of the supported work actions in it is to apply a threshold that applies to the entire database.

To support the management requirement, we use a WLM work action set that targets the whole database to categorize all the big reads for future analysis as shown in Table 6-5.

Table 6-5 Define work class and work class set

Work class set	Scope	Work class	Work class Type	Action
Large Work	Database	BIG_READS	READ	Queue READ query if <ul style="list-style-type: none"> ▶ TIMERONCOST > 100000.0 ▶ More than 10 concurrent DB coordinate activities

Example 6-17 shows the CREATE WORK CLASS SET statement.

Example 6-17 Create work class set

```
CREATE WORK CLASS SET toomanyreads ( WORK CLASS big_reads WORK TYPE READ FOR
TIMERONCOST FROM 100000.0 POSITION AT 1)
```

In Example 6-18, we create a work action set and specify the type of action DB2 WLM has to take when concurrent condition happens. In this case, we want to queue the high cost activity when the number of concurrent activities reaches more than 10.

Example 6-18 Create work action set to collect information and queue the work

```
CREATE WORK ACTION SET w1mdb FOR DATABASE USING WORK CLASS SET toomanyreads (
WORK ACTION action_iftoomany ON WORK CLASS big_reads WHEN
CONCURRENTDBCOORDACTIVITIES > 10 AND QUEUEDACTIVITIES > 10 COLLECT ACTIVITY
DATA WITH DETAILS CONTINUE ) DISABLE;
ALTER WORK ACTION SET w1mdb ENABLE;
```

6.5.3 Monitoring

We begin our monitoring from checking if the work classes, work action sets, and thresholds are set as we expected. Example 6-19 shows the work class created.

Example 6-19 Work class, work action definition and association

```
->db2 "SELECT WORKCLASSID,SUBSTR(workclassname,1,20)AS workclass,worktype FROM
SYSCAT.WORKCLASSES"
```

```
WORKCLASSID WORKCLASS          WORKTYPE
-----
          1 BIG_READS          2
```

```
1 record(s) selected.
```



```
->db2 "SELECT SUBSTR(actionname,1,20) AS actionname, actionid,
SUBSTR(actionsetname,1,20) AS actionsetname, SUBSTR(workclassname,1,20) AS
workclassname, enabled FROM syscat.workactions"
```

ACTIONNAME	ACTIONID	ACTIONSETNAME	WORKCLASSNAME	ENABLED
ACTION_IFTOOMANY	1	WLMDB	BIG_READS	Y

1 record(s) selected.

```
->db2 "SELECT SUBSTR(actionsetname,1,15) AS waset,SUBSTR(workclassetName,1,15)
wcset, SUBSTR(objectname,1,15) AS objectname, enabled FROM
syscat.workactionsets"
```

WASET	WCSET	OBJECTNAME	ENABLED
WLMDB	TOOMANYREADS	-	Y

1 record(s) selected.

We then check whether the thresholds were created, active, and the enforcement area. Example 6-20 lists the thresholds we created and they are all active. “U” under ORIGIN field means the threshold is created by a user, and “W” means the threshold is created through a work action.

Example 6-20 Check the threshold definitions and scope

```
-->db2 "SELECT SUBSTR(thresholdname,1,15) AS thresname,origin, enabled,
thresholdid, enforcement, queuesize, collectactdata FROM syscat.thresholds"
```

THRESNAME	ORIGIN	ENABLED	THRESHOLDID	ENFORCEMENT	QUEUESIZE	COLLECTACTDATA
TH_DB_TATIME	U	Y	1	D	0	D
SQL070913112834	W	Y	4	D	10	D

2 record(s) selected.

To monitor the activity, we check the number of concurrent queries running and queued activities as shown in Example 6-21. We collected the queue information using WLM table function WLM_GET_QUEUE_STATS. The output shows that none of connections or activities have been queued since the last reset.

Example 6-21 Collect the queue information

```
->db2 "SELECT SUBSTR(threshold_name, 1, 15) threshname, threshold_predicate,
threshold_domain, dbpartitionnum part, queue_size_top, queue_size_current,
queue_time_total, queue_assignments_total queue_assign FROM
TABLE(WLM_GET_QUEUE_STATS('', '', '', -1))"
```

THRESHNAME	THRESHOLD_PREDICATE	THRESHOLD_DOMAIN	PART	QUEUE_SIZE_TOP	QUEUE_SIZE_CURRENT	QUEUE_TIME_TOTAL	QUEUE_ASSIGN


```
SQL070913112834 CONCDBC      WA      0      0
0          0          0
```

1 record(s) selected.

6.5.4 Summary

For the monitoring data, we found that there's no activity was queued. The possible explanations include:

- ▶ There is no high cost query running on the system.
- ▶ There are high cost query but the number of concurrent jobs is under 10.

What can be done next are

- ▶ Re-evaluate whether the `TIMERONCOST > 100000.0` defined is too high. If not, no action has to be taken at this time.
- ▶ Reduce the number of concurrent queries from 10 to a lower accepted value

6.6 Stop user connections idle for more than 30 minutes

Connection idle time is the amount of time a connection is not working on the user request. When an application or a user connect to a system but is idle for a long time, it may create problem for the subsequent connections coming up to the system where the number of concurrent user limit has been in place. You can use DB2 WLM to stop applications or users that have connections idle for a certain amount of time. Care should be taken when implement this management control to not conflict with other management goals.

In this example, we demonstrate how to use DB2 WLM to stop the connections that have been idle for more than 30 minutes.

6.6.1 Identification

We use the workload and service classes defined in Appendix 6.3, "Consistent response time" on page 153 for this management requirement. No work class is required for this example.

6.6.2 Define controls

We create thresholds to check connection idle time (`CONNECTIONIDLETIME`) and terminate the connection if the limit is exceeded. This threshold has a

granularity of five minutes, so all values that you specify for the threshold are rounded to the nearest nonzero multiple of five minutes.

Example 6-22 shows the threshold definition.

Example 6-22 Create threshold

```
CREATE THRESHOLD th_act_time FOR DATABASE ACTIVITIES ENFORCEMENT DATABASE
ENABLE WHEN CONNECTIONIDLETIME > 30 MINUTES STOP EXECUTION;
```

Example 6-23 shows the thresholds created.

Example 6-23 List Active thresholds

```
->db2 "SELECT SUBSTR(thresholdname,1,15) AS thresname,origin, enabled,
thresholdid, enforcement, queuesize, collectactdata FROM syscat.thresholds"
```

THRESNAME	ORIGIN	ENABLED	THRESHOLDID	ENFORCEMENT	QUEUESIZE	COLLECTACTDATA
TH_DB_TATIME	U	Y	1	D		0 D
TH_ACT_TIME	U	Y	3	D		0 N
SQL070913112834	W	Y	4	D		10 D

3 record(s) selected.

6.6.3 Monitoring

To evaluate if this management requirements has been met, we collected information from event monitor THRESHOLDVIOLATIONS_THRESH_MON to check the stopped connections. Example 6-24 shows a sample output from threshold violations.

Example 6-24 Threshold violations

```
->db2 "SELECT SUBSTR(appl_id,1,40) AS appl_id,
SUBSTR(CHAR(coord_partition_num),1,4) AS part, threshold_action,
threshold_maxvalue,substr(threshold_predicate,1,28) AS threshold_predicate,
time_of_violation FROM thresholdviolations_thresh_mon ORDER BY 3 desc"
```

APPL_ID THRESHOLD_PREDICATE	PART TIME_OF_VIOLATION	THRESHOLD_ACTION	THRESHOLD_MAXVALUE
*NO.db2inst1.070913141132 ConnectionIdleTime	0 2007-09-13-09.41.56.000000	Stop	1800
*NO.db2inst1.070913145911 ConnectionIdleTime	0 2007-09-13-10.31.29.000000	Stop	1800
*NO.db2inst1.070913152356 ConnectionIdleTime	0 2007-09-13-10.56.07.000000	Stop	1800
9.50.23.43.2015.070913152901 ConnectionIdleTime	0 2007-09-13-10.59.25.000000	Stop	1800

*N0.db2inst1.070913151533	0	Stop	1800
ConnectionIdleTime	2007-09-13-11.07.21.000000		
9.50.23.43.2016.070913153940	0	Stop	1800
ConnectionIdleTime	2007-09-13-11.10.09.000000		
...			

When you find your threshold definition is not behaving as expected, it is worth to check the DB2CHECKCLIENTINTERVAL registry variable settings and make necessary changes. This variable specifies the frequency of TCP/IP client connection verifications. It permits early detection of client termination, instead of waiting until after the completion of the query. The default value for DB2CHECKCLIENTINTERVAL is 50.

6.6.4 Summary

From the monitoring data, we found the management requirement has been met.



WLM sample scenarios - Mixed OLTP and DSS environment

In many customer environments, the need to combine both Online Transaction Processing (OLTP) and Decision Support Systems (DSS) into a single environment is needed for more timely information and maintain a competitive advantage. One of the biggest challenges in a mixed OLTP and DSS environment is developing a proactive setup to allow them to coexist but not sacrificing throughput on either of them.

The setup environment used in this chapter shows how to identify, manage, and monitor such a mixed environment, using WLM. Again we have chosen to use the TCP-H benchmark data to build our environment. Simple queries were added to simulate the OLTP workloads.

7.1 Business objectives

The prime objectives typically seen in a mixed OLTP and DSS environment are:

- ▶ Separately identify the OLTP workloads from the DSS workloads.
- ▶ Insulate the OLTP workload performance from the DSS workloads.
- ▶ If needed, target a portion of the non-critical DSS workloads to be deferred until a later time, when the critical resources are no longer constrained.
- ▶ Maintain a consistent OLTP transaction response time.

The management of the availability of critical OLTP resources needs to be proactive in order to achieve a consistent OLTP transaction response time.

In this chapter we demonstrate how to use WLM to achieve these objectives.

Our business objectives for this exercise are stated as:

- ▶ Make sure our OLTP workload meets our expected throughput and is not unduly interrupted by the DSS workload.
- ▶ The DSS workload must also complete in a timely manner.
- ▶ The OLTP workload is an order entry system and requires an average response time of under 1 second.
- ▶ Our DSS workload consists of both ad-hoc queries and analysis reports.
- ▶ The ad-hoc queries have a higher priority than the analysis reports.
- ▶ All other workloads, such as administrative tasks, are only expected to be about 50 - 100 tasks a day.
- ▶ Database backups are part of the administrative workload. Several long running tasks are performed such as table space backups. These are not supposed to run during prime time shift (8:00 AM - 8:00 PM) and not interfere with the OLTP or DSS workloads.

7.2 Identify the work

Using our business objectives, we can identify our workloads and describe the management requirements.

- ▶ Administrative tasks:
 - Identified as all users in the group ID DB2ADM
 - Report when backups are run to make sure they don't run into the prime shift.

- ▶ OLTP:
 - Prime time shift is 8:00 AM to 8:00PM.
 - Runs as executable oltp.exe and runs as highest priority.
 - Report the average response times to make sure the are under one second on average.
- ▶ DSS (queries and reports)
 - Prime time shift is 8:00 AM to 8:00 PM.
 - Report workloads statistics for queries and reports separately.
- ▶ DSS queries
 - Identified as all users in group ID DSSGROUP.
 - Report the average response times to make sure that 90% are under five minutes.
- ▶ DSS reports
 - Runs as executable dss.exe.
 - Runs as lowest priority.
 - If necessary queue reports to limit their impact on OLTP and DSS queries.

Table 7-1 shows the worksheet identifying our workloads.

Table 7-1 Workload worksheet

Task	Business requirements	Identification	Action
Admin	Manages the database environment	groupid = DB2ADM	Report the times, duration, and frequency of tasks, such as backups
Batch	ETL must be complete prior to primetime shift	Loads and other ETL process using etl.exe or client userid = 'BATCH' and utility LOAD	Report the times, duration, and frequency of tasks
OLTP	Prime time shift is 8:00 AM - 8:00 PM. Queries need to complete in < 1 second	executable = oltp.exe	Assign highest priority, report average response times
DSS	Prime time 8:00 AM to 8:00 PM	Ad Hoc queries and reports run under dss.exe	Identify ad hoc separately from reports
_ DSS queries	Must complete 90% < 5 minutes	groupid = DSSGROUP	limit impact on OLTP. report average response times

Task	Business requirements	Identification	Action
_ DSS Analysis Reports	Must complete all reports daily	exec = dss.exe	limit impact on OLTP and Ad hoc queries

7.3 Manage the work

Given the worksheet details, we are now ready to build our configuration. Our build process is done in two steps. First we want to verify that our configuration is doing the tasks we want. Then we can adjust the configuration to queue or stop execution on the workloads that are impacting our business objectives.

Using the basic monitoring setup in Chapter 4, “Customizing the WLM execution environments” on page 61, we can add the additional requirements to our configuration.

7.3.1 Enabling the instance user ID to alter AIX priorities

The normal user rights given when creating an AIX user ID by default doesn't give the needed authority to allow the instance owner to adjust agent priorities. This capability must be added before any of agent priorities can be changed by DB2. Two additional capabilities are needed. The following command adds the needed AIX system privileges:

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE db2inst1
```

Where

- ▶ CAP_NUMA_ATTACH gives a process the ability to bind to specific resources.
- ▶ CAP_PROPAGATE permits all capabilities be inherited by child processes

Note: Root authority is needed to issue the **chuser** command. The capabilities are effective at the next user login. Therefore, the instance needs to be restarted from a new login session after the **chuser** command has been successfully run.

After the command has been successfully issued, the capabilities are displayed as shown in Example 7-1.

Example 7-1 AIX user capabilities

```
# lsuser db2inst1
```



```

db2inst1 id=1200 pgrp=db2adm groups=db2adm,staff,dasgrp home=/db2home/db2inst1
shell=/usr/bin/ksh login=true su=true rlogin=true daemon=true admin=false
sugroups=ALL admgroups= tpath=nosak ttys=ALL expires=0 auth1=SYSTEM
auth2=NONE umask=22 registry=files SYSTEM=compat logintimes= loginretries=0
pwdwarntime=0 account_locked=false minage=0 maxage=0 maxexpired=-1
minalpha=0 minother=0 mindiff=0 maxrepeats=8 minlen=8 histexpire=0
histsize=0 pwdchecks= dictionlist=
capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE fsize=-1 cpu=-1
data=-1 stack=-1 core=2097151 rss=65536 nofiles=-1 time_last_login=1189273673
time_last_unsuccessful_login=1188334437 tty_last_login=/dev/pts/0
tty_last_unsuccessful_login= host_last_login=9.26.92.65
host_last_unsuccessful_login=Clyde.itsosj.sanjose.ibm.com
unsuccessful_login_count=0 roles=

```

7.3.2 Creating the service classes definitions

Our first service class is our superclass, HIGHLVL. Under which we can create all our subclasses and indicate what level of information we want to capture.

- ▶ ADMINS

Since we don't plan on imposing any management rules, we collect only the minimal amount of data.

- ▶ BATCH

Here we are mainly interested in knowing the request execution times. This is collected using the COLLECT AGGREGATE REQUEST DATA BASE setting. This information is viewable in both SCSTATS and HISTOGRAMBIN tables.

- ▶ PROD_RPT

The reports that can be generated by this subclass can be quit resource intensive. Therefore, we want more information to assist in their management. The level of information we need is both at the aggregate request and activity levels. The second collection is added using the ALTER SERVICE CLASS. These two levels gives both the request execution times and averages about the activities in this service class to allow us to quickly analyze our workload. The COLLECT AGGREGATE ACTIVITY DATA EXTENDED was specified to add the average cost and arrival time information. We can use this additional data to understand how the PROD_RPT service class impacts our system.

- ▶ PROD_QRY

The ad-hoc queries needs to be slotted between our production reports and the OLTP transactions. In defining this subclass, we specify the same data collection as for our production reports. Additionally, we indicate that we want these queries to execute ahead of our production reports by setting AGENT PRIORITY -5. This is to satisfy our business requirement.

► OLTP

This transactions are very short lived and need to run at the highest priority in order to insure the business requirements of running in under one second can be maintained. The same data collection levels are used as for PROD_RPT and PROD_QRY.

Note: The aggregate collection data collection level has the least impact on our system. This enables us to collect useful information without unduly impacting our workloads. The anticipated impact to be between 5-10% for the service class.

Example 7-2 shows the service class definitions of our scenario.

Example 7-2 Service class definitions

```
--
-- Create superclass HIGHLVL
--
CREATE SERVICE CLASS highlvl DISABLE;
--
-- Create subclasses
--

CREATE SERVICE CLASS admins UNDER highlvl COLLECT AGGREGATE ACTIVITY DATA BASE
DISABLE;

CREATE SERVICE CLASS batch UNDER highlvl COLLECT AGGREGATE REQUEST DATA BASE
DISABLE;

ALTER SERVICE CLASS batch UNDER highlvl COLLECT AGGREGATE REQUEST DATA;

CREATE SERVICE CLASS prod_rpt UNDER highlvl COLLECT AGGREGATE ACTIVITY DATA
EXTENDED DISABLE;

ALTER SERVICE CLASS prod_rpt UNDER highlvl COLLECT AGGREGATE REQUEST DATA;

CREATE SERVICE CLASS prod_qry UNDER HIGHLVL agent PRIORITY -5 COLLECT AGGREGATE
ACTIVITY DATA EXTENDED DISABLE;

ALTER SERVICE CLASS prod_qry UNDER HIGHLVL COLLECT AGGREGATE REQUEST DATA;

CREATE SERVICE CLASS oltp UNDER highlvl AGENT PRIORITY -10 PREFETCH PRIORITY
HIGH COLLECT AGGREGATE ACTIVITY DATA EXTENDED DISABLE;

ALTER SERVICE CLASS oltp UNDER highlvl COLLECT AGGREGATE REQUEST DATA;
```

7.3.3 Creating the workload definitions

Our setup is similar to Chapter 4, so we only highlight the additional information. We need to identify our OLTP workloads so they are separated from the DSS workloads and can be properly monitored and managed.

- ▶ WL_OLTP Identifies our OLTP transactions as being run using the APPLNAME = oltp.exe
- ▶ The priorities have been changed to put WL_OLTP at the top so that it can be identified first and reduce the search time to identify our OLTP transactions.

Example 7-3 shows our workload definitions including the new additions and updates.

Example 7-3 Workload definitions

```
--
-- Create workloads
--
CREATE WORKLOAD w1_oltp APPLNAME ('oltp.exe') DISABLE SERVICE CLASS OLTP UNDER
highlvl POSITION AT 1;

CREATE WORKLOAD w1_batch CURRENT CLIENT_USERID ('BATCH') DISABLE SERVICE CLASS
BATCH UNDER highlvl POSITION AT 2;

CREATE WORKLOAD w1_prod_rpt APPLNAME ('dss.exe') DISABLE SERVICE CLASS
prod_rpt UNDER highlvl POSITION AT 3;

CREATE WORKLOAD w1_prod_qry SESSION_USER GROUP ('DSSGROUP') DISABLE SERVICE
CLASS prod_qry UNDER highlvl POSITION AT 4;

CREATE WORKLOAD w1_admin SESSION_USER GROUP ('DB2ADM') DISABLE SERVICE CLASS
admins UNDER highlvl POSITION AT 5;
```

Note: The workload definitions are searched in the order of their priority. To minimize search times, put the most critical workloads at the lowest priority number.

7.3.4 Finalizing the setup

Lastly, we grant the use to our workloads and enable the service classes and workloads as shown in Example 7-4.

Example 7-4 Granting permissions and enabling the service classes and workloads

```
--
```

```
-- Grant workload usage
--
GRANT USAGE ON WORKLOAD WL_ADMIN TO PUBLIC;
GRANT USAGE ON WORKLOAD WL_BATCH TO PUBLIC;
GRANT USAGE ON WORKLOAD WL_PROD_RPT TO PUBLIC;
GRANT USAGE ON WORKLOAD WL_PROD_QRY TO PUBLIC;
GRANT USAGE ON WORKLOAD WL_OLTP TO PUBLIC;
--
-- Enable service classes
--
ALTER SERVICE CLASS HIGHLVL ENABLE;
ALTER SERVICE CLASS ADMINS UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS BATCH UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS PROD_RPT UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS PROD_QRY UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS OLTP UNDER HIGHLVL ENABLE;
--
-- Enable workloads
--
ALTER WORKLOAD WL_ADMIN ENABLE;
ALTER WORKLOAD WL_BATCH ENABLE;
ALTER WORKLOAD WL_PROD_RPT ENABLE;
ALTER WORKLOAD WL_PROD_QRY ENABLE;
ALTER WORKLOAD WL_OLTP ENABLE;

COMMIT;
```

7.4 Monitoring the work

Having implemented our workload management strategy, we collect and monitor the data. We use data from the SCSTATS_BASIC_MON and HISTOGRAMBIN_BASIC_MON tables to demonstrate what monitoring data is used to measure the effectiveness of our WLM strategy. These two tables give us a good picture of how our workloads are behaving in the system. Additionally, we want to make sure our priorities are in effect and are being used.

7.4.1 Checking the agent priorities and prefetchers

With DB2 v9.5 being a threaded model, new functionality has been incorporated to allow us to look at the details of the threads. Use the following command to obtain service class details:

```
db2pd -db w1mdb -serviceclasses
```

Example 7-5 show the service class definitions and that the OLTP already is being used (see the highlighted lines)

Example 7-5 Using db2pd -service classes

```

...
Service Class Name      = PROD_QRY
Service Class ID       = 19
Service Class Type     = Service Subclass
Parent Superclass ID   = 14
Service Class State    = Enabled
Agent Priority        = -5
Prefetch Priority      = Default
Outbound Correlator    = _HighLevel.Prod_QRY
Collect Activity Opt   = None
Collect Aggr Activity Opt = Extended
Collect Aggr Request Opt = Base
Act Lifetime Histogram Template ID      = 1
Act Queue Time Histogram Template ID    = 1
Act Execute Time Histogram Template ID   = 1
Act Estimated Cost Histogram Template ID = 1
Act Interarrival Time Histogram Template ID = 1
Request Execute Time Histogram Template ID = 1

Access Count           = 0
Last Stats Reset Time = 09/12/2007 12:51:02.000000
Activities HWM         = 0
Activities Completed   = 0
Activities Rejected    = 0
Activities Aborted     = 0

Associated Agents:
EDU ID      AppHandl [nod-index]  WL ID      WLO ID      UOW ID
Activity ID

Associated Non-agent threads:
PID          TID              Thread Name

Service Class Name      = OLTP
Service Class ID       = 20
Service Class Type     = Service Subclass
Parent Superclass ID   = 14
Service Class State    = Enabled
Agent Priority        = -10
Prefetch Priority    = High
Outbound Correlator    = None
Collect Activity Opt   = None
Collect Aggr Activity Opt = Extended

```

```

Collect Aggr Request Opt = Base
Act Lifetime Histogram Template ID      = 1
Act Queue Time Histogram Template ID    = 1
Act Execute Time Histogram Template ID  = 1
Act Estimated Cost Histogram Template ID = 1
Act Interarrival Time Histogram Template ID = 1
Request Execute Time Histogram Template ID = 1

Access Count          = 5
Last Stats Reset Time = 09/12/2007 12:51:02.000000
Activities HWM        = 10
Activities Completed  = 57
Activities Rejected   = 0
Activities Aborted    = 0

```

Associated Agents:

EDU ID	AppHandl	[nod-index]	WL ID	WLO ID	UOW ID
Activity ID					
37671158153220	16372	[000-16372]	3	92	7
1					
90838558310404	16373	[000-16373]	3	93	5
1					
35407710388228	16374	[000-16374]	3	94	6
1					
339061898215428	16378	[000-16378]	3	98	7
1					
292723496058884	16379	[000-16379]	3	99	0
0					

To check what priorities agents are using, run the command

```
db2pd -db w1mdb -age
```

Example 7-6 shows that we are running eight OLTP transactions and they are all running at a priority of -10.

Example 7-6 db2pd agent priorities

```
->db2pd -age -dbp 1 | grep -v Pooled
```

```
Database Partition 1 -- Active -- Up 0 days 19:09:56
```

Agents:

```

Current agents:      10
Idle agents:         0
Active coord agents: 1
Active agents total: 10

```

Address	AppHandl	[nod-index]	AgentEUID	Priority	Type	State
ClientPid	Userid	ClientNm	Rowsread	Rowswrtn	LkTmOt	DBName
0x07800000023ACC0	67157	[001-01621]	9879	0		Coord
Inst-Active	2728214	dssuser	db2stmm	0	0	NotSet WLMDB
0x07800000023EAE0	14189	[000-14189]	5032	0		SubAgent
Inst-Active	2928	PESRVUSR	db2evmt_	0	16592	3
WLMDB						
0x0780000000942A00	17010	[000-17010]	10533	-10		SubAgent
Inst-Active	2199916	dssuser	oltp.exe	2653	0	NotSet WLMDB
0x0780000000944180	17011	[000-17011]	10790	-10		SubAgent
Inst-Active	1679658	dssuser	oltp.exe	2653	0	NotSet WLMDB
0x0780000000940080	17008	[000-17008]	10019	-10		SubAgent
Inst-Active	2666978	dssuser	oltp.exe	2653	0	NotSet WLMDB
0x0780000000941540	17009	[000-17009]	3113	-10		SubAgent
Inst-Active	606550	dssuser	oltp.exe	2653	0	NotSet WLMDB
0x0780000000945B40	17013	[000-17013]	11051	-10		SubAgent
Inst-Active	1626264	dssuser	oltp.exe	2653	0	NotSet WLMDB
0x0780000000947280	17015	[000-17015]	11309	-10		SubAgent
Inst-Active	237574	dssuser	oltp.exe	2653	0	NotSet WLMDB
0x07800000009489C0	17016	[000-17016]	11566	-10		SubAgent
Inst-Active	2113846	dssuser	oltp.exe	2653	0	NotSet WLMDB
0x078000000023D620	17014	[000-17014]	10289	-10		SubAgent
Inst-Active	2146474	dssuser	oltp.exe	2653	0	NotSet WLMDB
....						

Note: The agent threads are set to a priority equal to the default priority, plus the value set when the next activity begins.

In UNIX and Linux systems, the valid values are -20 to +20 (a negative value indicates a higher relative priority). In Windows-based platforms, the valid values are -6 to +6 (a negative value indicates a lower relative priority)

Use the following steps to check the prefetchers:

1. Switch to a data partition

```
export DB2NODE=1
db2 terminate
```

2. Obtain the process ID of db2sysc of the partiton using the following command:

```
ps -ef | grep db2sysc
```

The process ID for db2sysc 1 is 987302.

3. Show the threads for the db2sysc process 987302

```
ps -m -o THREAD -p 987302
```

Example 7-7 shows the output from these two commands while a prefetcher was actively running.

Example 7-7 Output from ps -m command for prefetchers

```
->ps -ef | grep db2sysc | grep -v grep
db2inst1 987302 1261848 0 18:25:39 - 63:06 db2sysc 1
db2inst1 1130766 1233344 0 18:25:40 - 61:16 db2sysc 2
db2inst1 1270196 614908 0 18:25:39 - 45:18 db2sysc 0
...
->ps -m -o THREAD -p 987302
  USER      PID      PPID      TID ST  CP PRI SC   WCHAN      F      TT BND  COMMAND
db2inst1 987302 1261848 - A   0 60 41      *    40401 - - - db2sysc 1
- - - - 1433801 S   0 60 1      -    400400 - - -
- - - - 1880209 S   0 60 1      -    400400 - - -
- - - - 2044119 S   0 60 1 f1000004e0108c00 410400 - - -
- - - - 2068605 S   0 60 1      -    400400 - - -
- - - - 2170933 S   0 60 1 f100011022c23230 410400 - - -
- - - - 2265111 S   0 60 1 f100000500108400 410400 - - -
- - - - 2543691 S   0 60 1      -    418400 - - -
- - - - 3125337 Z   0 60 1      -    c00001 - - -
- - - - 3194901 S   0 60 1 f1000004f010bc00 410400 - - -
- - - - 3444955 S   0 60 1 f1000004d010ee00 410400 - - -
- - - - 3690681 S   0 60 1 f10000050010f200 410400 - - -
- - - - 3936299 S   0 60 1      -    400400 - - -
- - - - 4046931 S   0 60 1      -    418400 - - -
- - - - 4132945 S   0 60 1      -    418400 - - -
- - - - 4448321 S   0 60 1 f10000050010f600 410400 - - -
```

4. List the EDU's for the prefetchers

```
db2pd -db w1mdb -edu -dbp 1
```

Example 7-8 shows the db2pfchr data.

Example 7-8 Output from db2pd -edu for prefetchers

```
->db2pd -db w1mdb -edu -dbp1 | egrep "EDU Name|db2pfchr"
```

EDU ID	TID	Kernel TID	EDU Name	USR	SYS
9658		2941051	5906937		0.001285
0.000531			db2pfchr (WLMDB) 1		
9401	9401	2265111	db2pfchr (WLMDB) 1	0.001099	0.000597
9144	9144	4682099	db2pfchr (WLMDB) 1	0.001664	0.001061
8887	8887	4853817	db2pfchr (WLMDB) 1	0.002559	0.001224
8630	8630	4166005	db2pfchr (WLMDB) 1	0.001728	0.001077
8373	8373	3194901	db2pfchr (WLMDB) 1	0.001910	0.001030
8116	8116	2793797	db2pfchr (WLMDB) 1	0.002762	0.001750
7859	7859	5435469	db2pfchr (WLMDB) 1	0.017083	0.012175
7602	7602	5853695	db2pfchr (WLMDB) 1	0.048361	0.039011
7345	7345	3690681	db2pfchr (WLMDB) 1	0.058649	0.041682
7088	7088	5816601	db2pfchr (WLMDB) 1	0.057663	0.044016
6831	6831	4448321	db2pfchr (WLMDB) 1	0.065778	0.039444
6574	6574	3068261	db2pfchr (WLMDB) 1	0.066206	0.045546
6317	6317	3444955	db2pfchr (WLMDB) 1	0.077666	0.054751
6060	6060	6455555	db2pfchr (WLMDB) 1	0.075359	0.048456

7.4.2 Monitoring and analyzing the service classes

To begin with, let us get a high level view of our workloads by looking at the HISTOGRAMBIN_BASIC_MON table. Using a query as shown in Example 7-9, we can see histogram data and plot graphs that give us an overview of how our system is performing.

Example 7-9 Query histogram_basic_mon table

```

WITH hist_reg ( bin, subclass, nbr_in_bin) AS
  (SELECT bin_top,
         SUBSTR(service_subclass,1,15),
         NUMBER_IN_BIN
   FROM   histograms
   WHERE  HISTOGRAM_TYPE = 'ReqExecTime'
   ORDER BY 1,2
  ),
hist_colife ( bin, subclass, nbr_in_bin) AS
  (SELECT BIN_TOP,
         SUBSTR(Service_subclass,1,15),
         number_in_bin,
   FROM   histograms
   WHERE  histogram_type = 'CoordActLifetime'
   ORDER BY 1,2
  ),
hist_coexec ( bin, subclass, nbr_in_bin) as
  (SELECT bin_top,
         SUBSTR(SERVICE_SUBCLASS,1,15),
         number_in_bin
   FROM   histograms
   WHERE  histogram_type = 'CoordActExecTime'
   ORDER BY 1,2
  ),
hist_coarrive ( bin, subclass, nbr_in_bin) AS
  (SELECT bin_top,
         SUBSTR(service_subclass,1,15),
         number_in_bin
   FROM   histograms
   WHERE  histogram_type = 'CoordActInterArrivalTime'
   ORDER BY 1,2
  ),
hist_cec ( bin, subclass, nbr_in_bin) AS
  (SELECT BIN_TOP,
         SUBSTR(service_subclass,1,15),
         number_in_bin
   FROM   histograms

```

```
WHERE histogram_type = 'CoordActEstCost'
ORDER BY 1,2
)
SELECT r.bin, r.subclass,
       r.nbr_in_bin as ReqExecTime,
       l.nbr_in_bin as CoordActLifetime,
       e.nbr_in_bin as CoordActExecTime,
       a.nbr_in_bin as CoordActInterArrivalTime,
       c.nbr_in_bin as CoordActEstCost
FROM hist_reg r, hist_cec c, hist_colife l, hist_coexec e,
     hist_coarrive a
WHERE r.bin=c.bin
AND r.bin=l.bin
AND r.bin=e.bin
AND r.bin=a.bin
AND r.subclass = c.subclass
AND r.subclass = l.subclass
AND r.subclass = e.subclass
AND r.subclass = a.subclass

ORDER BY 2,1 ;S
```

Request execution time

The first chart (Figure 7-1) we want to examine is the request execution times. This gives us a picture of “how are we doing against our execution time objectives”.

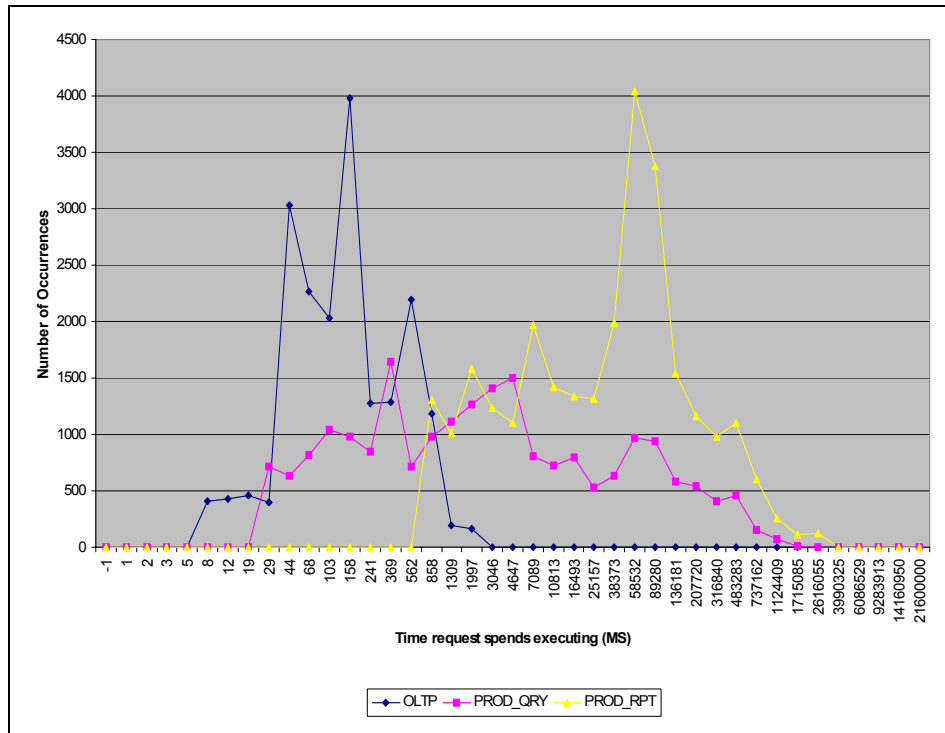


Figure 7-1 Request execution time

Here we see where our execution times fall in relationship to our objectives. Our OLTP is running between .008 and 3.046 seconds. Our objective is to run all of our OLTP transactions under 1 second but we can see that some transactions are outside of our objective. This objective will need further analysis.

The production queries, our second highest priority are running between .029 and 1717.085 seconds (28.6 minutes). Our objective is to complete 95% in under 5 minutes. We completed a total of 21,286 transactions and 21,057 completed in under 5 minutes. Our objective was 95%, we achieved 98.92%. Objective met!

As for our production reports, we ran 27,572 reports during the period. All were completed but did they interfere with any of our objectives? For this we need more information.

Workload arrival rate

Next we turn our attention to the arrival rates of our workloads as illustrated in Figure 7-2.

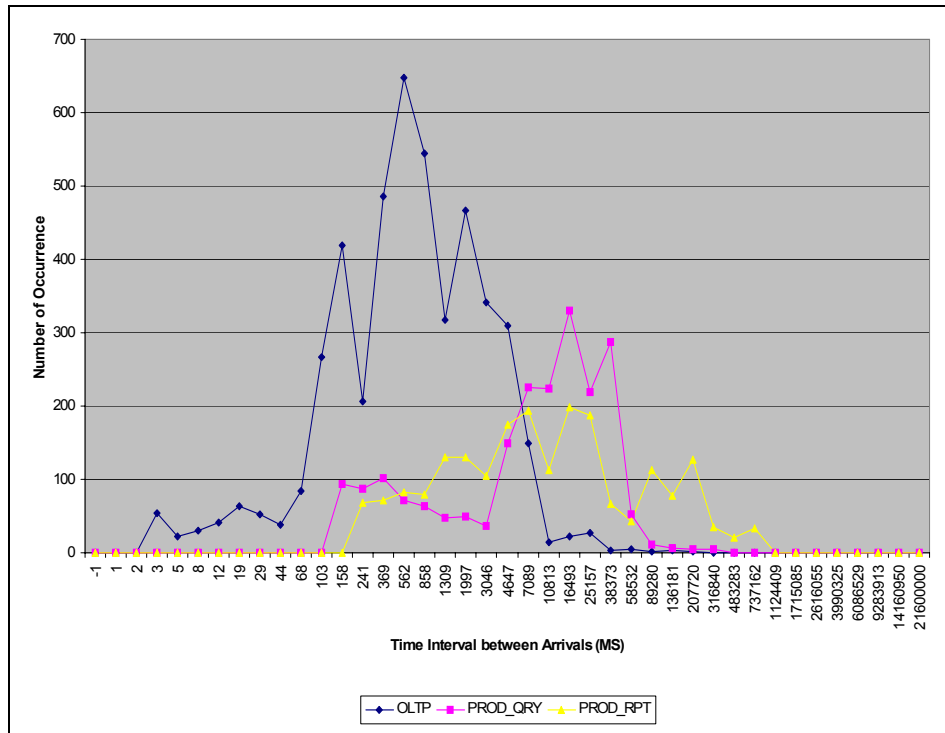


Figure 7-2 Workload arrival rate

Here we see how “fast” workloads are arriving in our system. Are we overloading our system at critical times? That is the question we want to answer. Our focus is on the Production reports (PROD_RPT). From Figure 7-2, we can see the PROD_RPTS peak at nearly 200 arrivals over a 25 second interval. Is this the expected arrival rate? Does the high arrival rate of PROD_RPTS interfere with our OLTP transactions? We need to examine when the transactions arrive by looking at the aggregate service class statistics (SCSTATS_BASIC_MON) table.

Request execution time in a time frame

From Figure 7-3, we can focus on the execution times for specific time periods. Here we can see when the Request execution times for OLTP miss the targeted business objectives. During the time frame of 14:00 to 17:30, the OLTP request times were consistently out of our targeted business objectives of less than one second execution time.

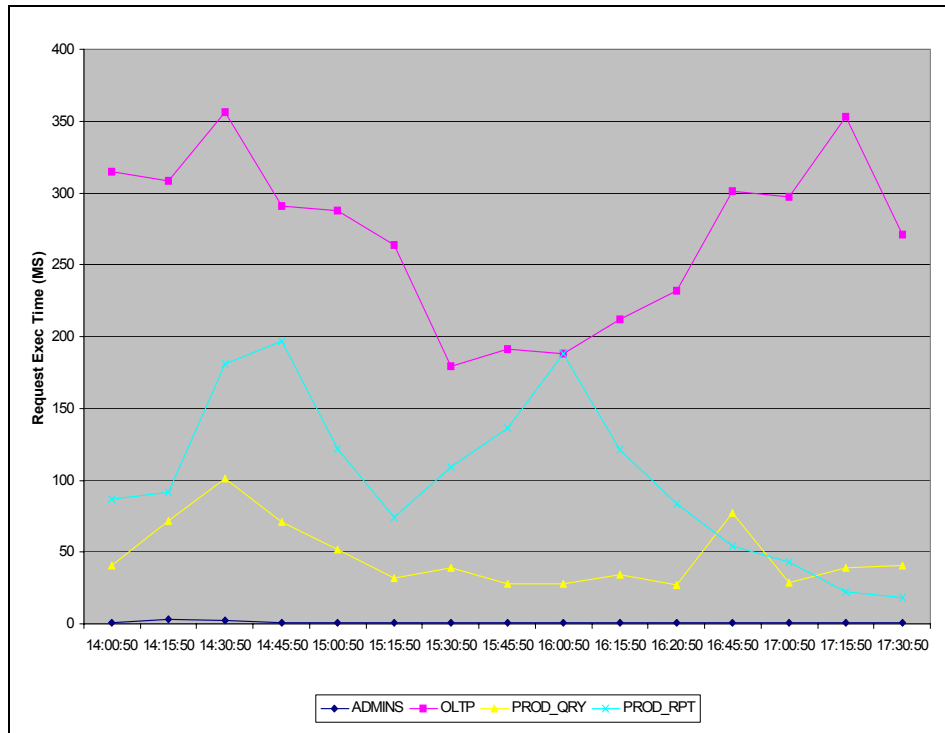


Figure 7-3 Request execution time in a specific time frame

Next we need to look for the cause of the excessive OLTP execution times. From the chart in Figure 7-3, we have a clue. The PROD_RPT executions times are also quite long. What else can we analyze to give us a clearer picture?

In Figure 7-4 on page 190, we look at the arrival times for workloads during the same time frame. Here we see a lot of workloads arriving at the same time. Notice the larger number of PROD_RPT workloads.

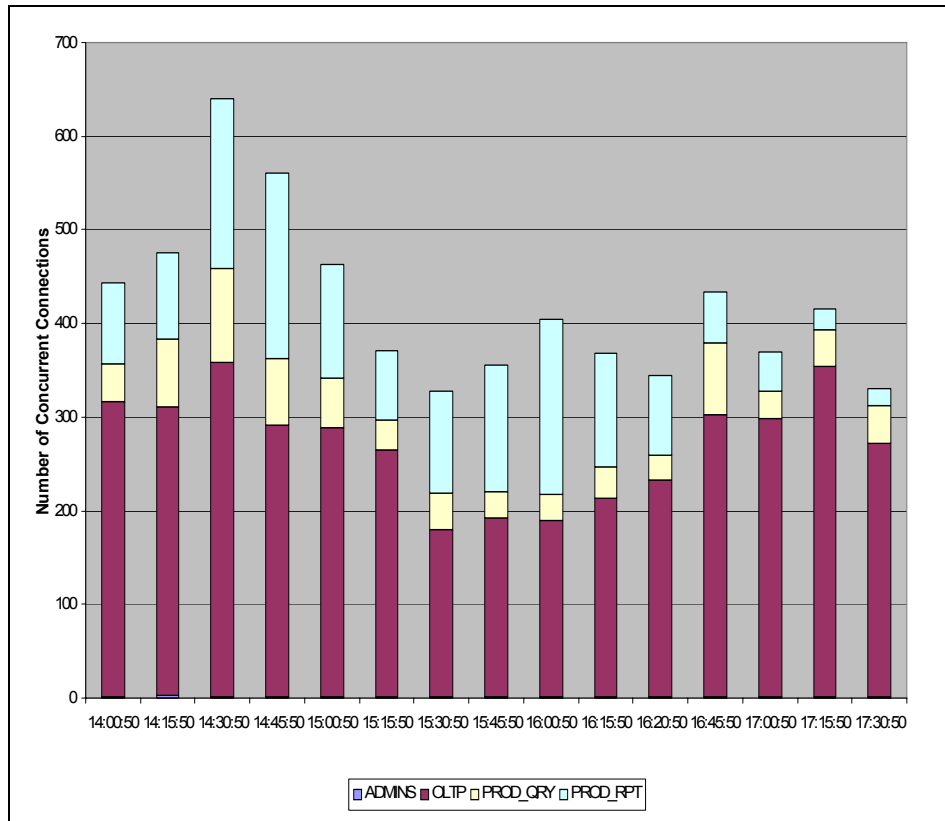


Figure 7-4 Workload arrival time in a specific time frame

Workload cost

Now we want to learn more about the PROD_RPTS for this time frame. In Figure 7-5, we focus on the costs of these reports.

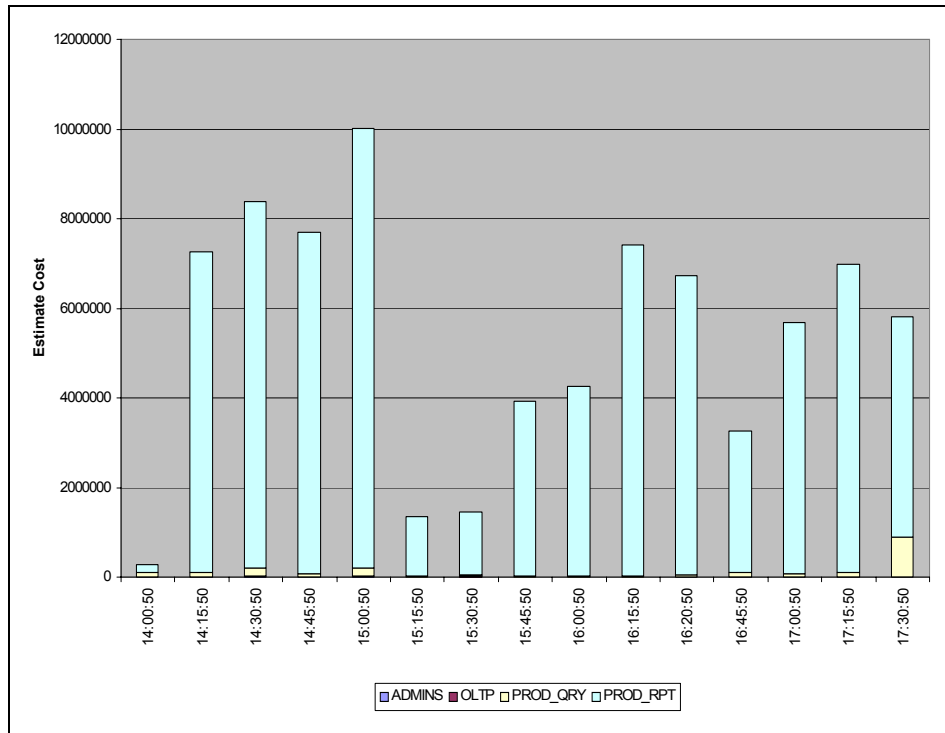


Figure 7-5 Estimate cost of workloads

7.5 Summary

Looking back at our earlier histogram chart, we see the PROD_RPTS are running near the top of the costs reported in the histograms. To summarize our analysis so far, we see a lot of PROD_RPTS running at the same time and these reports are the most expensive reports. At the same time, we see our OLTP transactions begin to run longer and miss our targeted objectives. The cause appears to be the long running and expensive reports.

We now need to develop a management strategy to limit the number of expensive reports from running when the number of concurrent coordinators is also high. This will give us a proactive WLM strategy that only is enforced during peak transaction periods. The approach we want to take is a global approach since the PROD_RPTS costs are much higher than any other “known” workload.

To identify this new WLM requirement, we develop a management strategy shown in Example 7-10. We implement a global threshold to limit the concurrent coordinators based on query costs and concurrent workloads.

This additional WLM strategy states that when the number of concurrent database coordinators is greater than 400, we want to queue any workloads that have a timeron cost equal or greater than 2,000,000. No upper bound or maximum number of queued workloads is set at this time. With more data and analysis, we might want to limit the size of the queue.

Example 7-10 Global Threshold to limit based on query costs and concurrent workloads

```
CREATE WORK CLASS SET large_work
  (WORK CLASS big_costs WORK TYPE READ
   FOR TIMERONCOST FROM 2000000 TO UNBOUNDED) ;

CREATE WORK ACTION SET longrun FOR DATABASE
  USING WORK CLASS SET large_work
  (WORK ACTION TOO_MANY ON WORK CLASS BIG_COSTS
   WHEN CONCURRENTDBCOORDACTIVITIES > 400
   COLLECT ACTIVITY DATA CONTINUE)
  DISABLE ;
```



AIX Workload Manager considerations

There are two different kinds of workload management solutions available, Operating system (OS) workload management solutions and application level workload management solutions. DB2 Workload Manager (DB2 WLM) is an example of later, AIX Workload manager (AIX WLM) is an example of OS level workload management system.

DB2 9.5 Workload Manager supports AIX WLM. Integrating DB2 WLM with AIX WLM provides you even more capability in managing and controlling the workloads and resource on you database system.

In this chapter we discuss the following topics:

- ▶ AIX WLM overview
- ▶ Using DB2 WLM and AIX WLM

8.1 AIX WLM overview

AIX WLM provides the capability of isolating applications, including DB2, with very different system behaviors. Based on the business objectives, AIX WLM can allocate CPU, physical memory, and I/O bandwidth to the classified applications.

The database system is usually considered as the most important application running on the server. AIX WLM is an ideal solution for protecting the database applications from being interfered by the other applications running on the same server.

8.1.1 Service classes

Conceptually, AIX WLM is similar to DB2 WLM. AIX WLM has workloads and two level hierarchical service classes. In addition to the five predefined service classes, AIX WLM can have as much as 27 user defined superclasses. Each superclass can have 10 user defined subclasses and two predefined sub classes. No process can belong to superclass only but not to subclass. Every process in the system is mapped to the predefined Default subclass unless it is explicitly defined to other subclass by the administrator.

Superclasses

The predefined AIX WLM superclasses are as follows:

- ▶ Default superclass
All user processes belong to this superclass unless explicitly defined to map to other superclass.
- ▶ System superclass
This is default superclass for all privileged processes owned by root.
- ▶ Shared superclass
This superclass receives all memory pages that are shared by processes which belong to more than one superclass.
- ▶ Unclassified superclass
When AIX WLM starts, all processes and memory pages are assigned to certain superclass. The memory pages which cannot be tied to any specific process will be mapped to this superclass.
- ▶ Unmanaged superclass
This class is reserved for memory pages which are unmanaged by AIX WLM. This superclass does not have any shares or limits for any resources. This superclass does not have any sub classes. No processes will be assigned to this superclass.

Figure 8-1 illustrates how resources are assigned with default AIX WLM configuration.

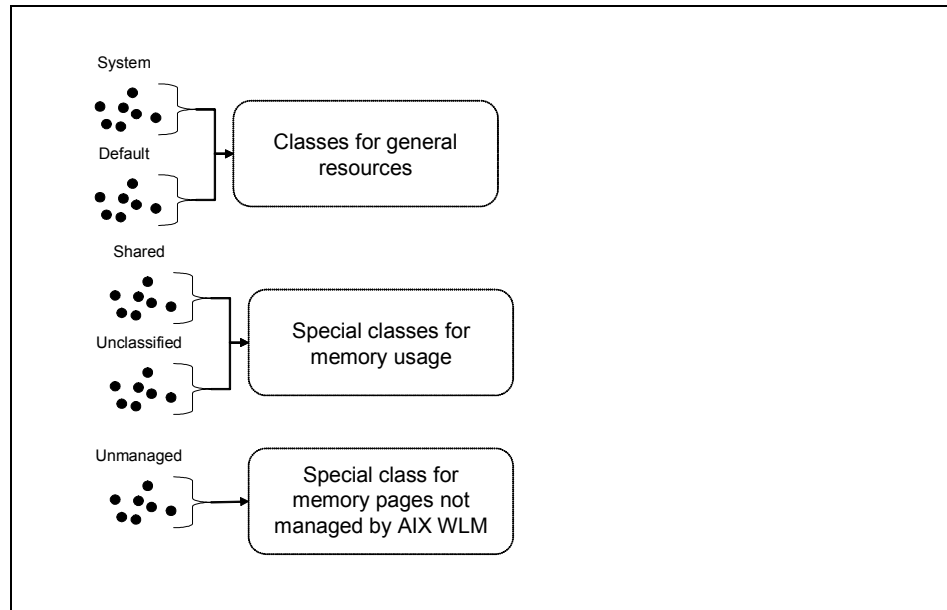


Figure 8-1 AIX WLM default service classes

Subclasses

Subclass can have only one superclass. There exists two predefined sub classes:

- ▶ **Default subclass**
All processes which are not defined to any other subclass will be assigned to Default subclass.
- ▶ **Shared subclass**
Like Shared superclass, Shared subclass contains all the memory pages which contain processes belonging to more than one subclass under same superclass.

Tiers

Tier defines the importance of class. There are 10 levels of tiers starting from 0 to 9. Tier 0 is the most important level and has highest priority. If the classes in tier level 0 takes up all resources, tier level 1 does not get any resources. If your tier level 0 and tier level 1 classes are taking all system resources, there will be nothing left for tier 2 and so on. If no tier value is defined, the default value 0 will be used. Tier is defined both to superclasses and subclasses.

General class attributes

A class can have the following attributes:

- ▶ Class name: Up to 16 characters long.
- ▶ Tier: Values from 0 to 9. Tier value enables you to prioritize groups of classes. The default and highest priority value is 0.
- ▶ Inheritance: Defines whether child process or thread inherits its parents classes assignment.
- ▶ Authuser and authgroup: This is used give user or group right to manually assign process to certain class. These attributes can be used only for superclasses.
- ▶ Adminuser and admingroup: These are used to delegate administration rights to certain user ID or group. These attributes can be used only for superclasses.
- ▶ Recourse set: This limits recourses which the class is entitled to.
- ▶ Localsmh: Specifies whether memory segments that are accessed by processes in different classes remain local to the class they were initially assigned to, or if they go to the Shared class.

Class limits and shares for CPU, memory, or disk I/O resource

Shares defines the proportion of a resource the process in a service class can get. With shares, you can allocate a certain amount of CPU, memory, or I/O resources to each service class. The resources will be allocated to services class relatively depending on the amount of total shares and the amount of shares the service class is having itself. For instance, if we have total 1000 CPU shares for all superclasses and individual superclass db2Def has 400 CPU shares, db2Def will get 40% of all CPU resources. If we increase the amount of total CPU shares for all superclasses to 2000, with the 400 CPU shares db2Def has, it would have anymore 20% of all CPU resources.

The resource share principle applies to memory and disk I/O resources as well.

You can also define resource limits by percentage for different service classes. The available limits are:

- ▶ min
Specifies the minimum percentage of a resource that will always be granted to a service class. Default is 0.
- ▶ softmax
Specifies maximum percentage of the resource that can be assigned to a service class when there is contention. If there is no contention, service class can get more resources than what is specified by this attribute. If there is

contention, it is not guaranteed that the service class will get the percentage of the resource specified by this attribute. The default is 100.

- ▶ **hardmax**
Specifies maximum percentage of the resource assigned for a service class when there is no contention. The default is 100.

Class assignment

There are two ways to classify processes to different service classes: manual and automatic. When you want to assign processes manually to certain service classes, you can do it with command **wlmsign <PID>**.

Automatic classification is done by following certain rules that are defined after service class is defined. These rules can be defined in class assignment rules which can contain following attributes:

- ▶ **Order of the rule:** This is any number which defines the order of rules.
- ▶ **Class name:** This defines to which class processes or thread will be mapped.
- ▶ **User and Group:** These define by which user ID or group ID a process is owned.
- ▶ **Application:** This define full path name of the application and can contain wild cards.
- ▶ **Tag:** This is a label for identifying processes and threads, that will be assigned by AIX WLM API call. DB2 uses tags to map DB2 WLM service classes to certain AIX WLM service classes.

You can access class assignment rules either through smitty, or by editing special rules file. */etc/wlm/current/rules* is the rules file for all super classes. For subclasses rules files are located under corresponding superclass directory. All superclass directories for running WLM configuration can be found under directory */etc/wlm/current/<superclass name>*.

8.1.2 Monitoring

There are many AIX WLM aware OS monitoring tools to monitor AIX WLM that includes:

- ▶ **wlmstat** is similar to the use of **vmstat** or **iostat**.
- ▶ **nmon** has AIX WLM enhancements.
- ▶ **topas** is AIX WLM aware
- ▶ **ps** is AIX WLM aware: **ps -m -o THREAD,class -p <process id>**

8.1.3 Configuring AIX WLM

All AIX WLM related configuration files are located in `/etc/wlm` directory. The subdirectories contain different configuration sets as show in Example 8-1.

Example 8-1 Contents of /etc/wlm

```
# ls -la /etc/wlm
total 24
drwxr-xr-x  7 root    system      256 Aug 28 18:11 .
drwxr-xr-x 19 root    system      8192 Aug 27 16:35 ..
-----  1 root    system        0 Aug 21 12:35 .lock
dr-xr-sr-x  3 root    system      256 Aug 28 18:11 .running
lrwxrwxrwx  1 root    system        18 Aug 28 18:11 current ->
/etc/wlm/standard
drwxr-xr-x  3 root    system      256 Aug 23 15:13 inventory
drwxr-xr-x  2 root    system     4096 Aug 23 11:33 standard
drwxr-xr-x  2 root    system      256 Dec 05 2004 template
```

In this section, we describe how to configure AIX WLM by setting up a simple WLM configuration.

You can use `/etc/wlm/template` as a template for setting up your own AIX WLM configuration. The steps are:

1. Copy the template to your desired directory

```
cp -r /etc/wlm/template /etc/wlm/testenv
```
2. Set new configuration set as current configuration

```
wlmcntrl -d testenv
```
3. Create superclass and subclasses

AIX WLM can be configured using `smitty` or by commands and editing WLM specific configuration files. Example 8-2 shows commands to create super and subclasses. In this example, we created superclass `HighLevel` and gave it 100 CPU shares. Under superclass `HighLevel` we created two subclasses: `Prod` with 60 CPU shares and `Utils` with 40 CPU shares.

Example 8-2 Creating AIX WLM service classes

```
mkclass -a inheritance=no -c shares=100 HighLevel
mkclass -a inheritance=no -c shares=60 HighLevel.Prod
mkclass -a inheritance=no -c shares=40 HighLevel.Utils
```

4. Map application to superclass

You can map application to superclass either by using smitty or by editing specific rules file. If you prefer to use smitty, use command **smitty wlm** to start it and navigate to “Class assignment rules”.

In this example, we have two applications:

- app1_app:
The binaries are in /opt/App1/bin. Application app1_app has a high service level agreement (SLA) which we want to achieve.
- app1_batch:
The binaries are in /opt/App1/batch/bin. app1_batch collects the performance statistics of app1_app. Even though app1_batch is important, we do not want it to have any performance impact on app1_app.

We define our control rules by editing /etc/wlm/testenv/rules as shown in Example 8-3. We mapped both applications in directories /opt/App1/bin and /opt/App1/batch/bin to superclass HighLevel.

Example 8-3 Superclass mapping

```
* IBM_PROLOG_BEGIN_TAG
* This is an automatically generated prolog.
*
* bos530 src/bos/etc/wlm/rules 1.2
*
* Licensed Materials - Property of IBM
*
* (C) COPYRIGHT International Business Machines Corp. 1999,2002
* All Rights Reserved
*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*
* IBM_PROLOG_END_TAG
* class resvd user group application type tag
System - root - - - -
HighLevel - - - - /opt/App1/bin/* - -
HighLevel - - - - /opt/App1/batch/bin/* - -
Default - - - - - -
```

5. Map application to subclass

The rules file for subclasses running under superclass HighLevel is /etc/wlm/testnev/HighLevel/rules. This file is always created when you create the superclass. We edit this file as show in Example 8-4.

Example 8-4 Subclass mapping

```
* class resvd user group application type tag
```

Prod	-	-	-	/opt/App1/bin/*	-	-
Utils	-	-	-	/opt/App1/batch/bin/*	-	-

6. Update configuration settings

Once the service classes and mapping rules are defined, use the following command to update the AIX WLM running configuration:

```
wlmcntrl -u
```

Our intention was to ensure that the production application app1_app always gets priority over the utility app1_batch. We had set subclass HighLevel.Prod for app1_app and subclass HighLevel.Util for app1_batch. We gave the priority simply by assigning 60 CPU shares for app1_app and 40 CPU shares for app1_batch of 100 shares total.

Figure 8-2 shows that our system reflects this setting. In the picture we see that by setting shares 60/40, subclass HighLevel.Prod gets prioritized over subclass HighLevel.Util. When using shares, the average CPU time inside the superclass Highlevel will be 60% for HighLevel.Prod and 40% for HighLevel.Util.

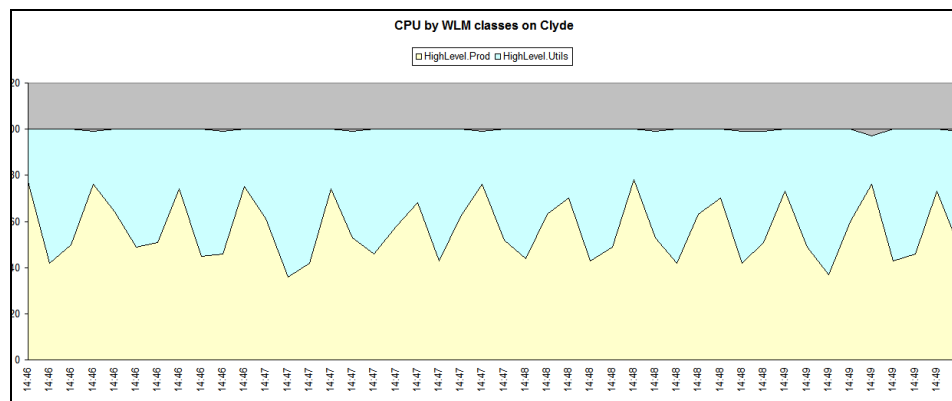


Figure 8-2 CPU by service class

Because we did not define any min, max, or hardmax values for our service classes, there was no specific CPU usage percentage limits to be forced upon both service classes. When we look average data at same time period, we find that the average CPU for HighLevel. Prod was 60% and for HighLevel.Util 40% as illustrated in Figure 8-3.

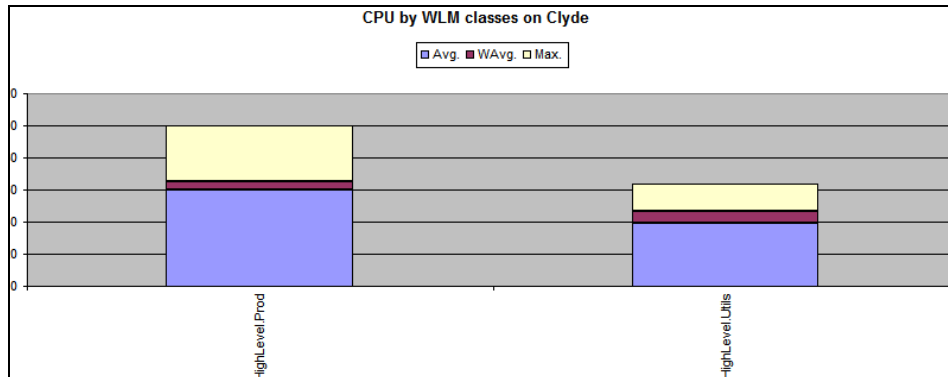


Figure 8-3 Average CPU time by service classes

We ran same test without defining AIX WLM CPU shares. As you can see from Figure 8-4, the result looks a lot different than it was when CPU shares were defined.

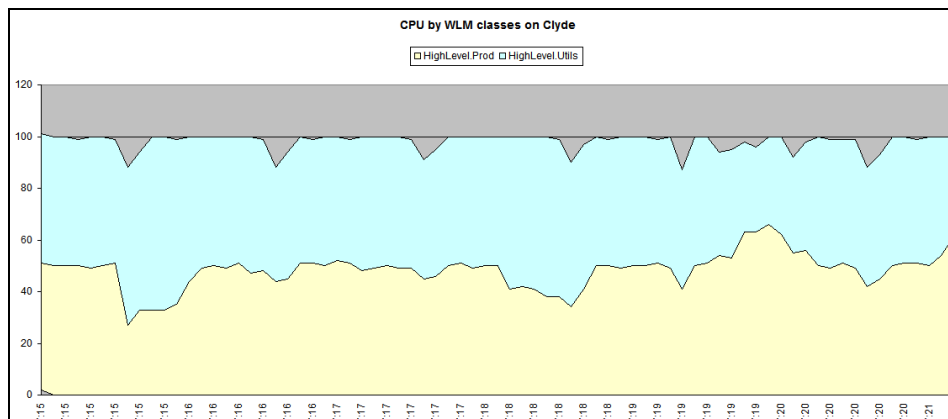


Figure 8-4 CPU by service class without CPU shares

There is major difference between average values as well. See Figure 8-5 on page 202. As you can see, on average there was no difference on consumed CPU time between the applications. In fact there was no guarantee that app1_app could get resources it needed. This could have led to poor service quality and violation of the business objective.

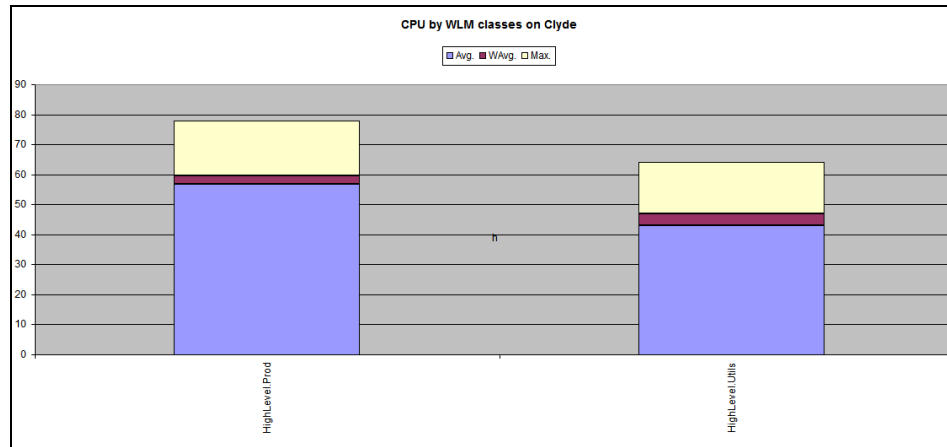


Figure 8-5 Average CPU time by service classes without CPU shares

8.2 Using DB2 WLM and AIX WLM

In this section we discuss how to integrate DB2 WLM with AIX WLM. We introduce the general guidelines in planning and designing an integrated environment. We provide useful examples and discuss monitoring tools.

8.2.1 General guidelines

For DB2 9.5, the AIX WLM support for DB2 WLM is CPU resource allocation only. If you want to prioritize I/O resource usage, you can set prefetch priority on DB2 WLM service class. You can set prefetch priority in DB2 WLM and define CPU prioritization through AIX WLM simultaneously. When you integrate DB2 WLM with AIX WLM, you can not set AGENT PRIORITY for DB2 service classes to be anything else but “Default”, since it will prevent setting outbound correlator to be used with operating system WLM.

AIX WLM is capable of using application tags to map processes and threads to desired service classes. You define application tag through DB2 WLM configuration by setting outbound correlator for your service classes. After integrated your WLM environments, AIX can manage the CPU utilization for DB2 service classes.

8.2.2 Mapping schemes

Before you begin using DB2 WLM and AIX WLM, you need to think how to map different DB2 service classes with AIX WLM service classes. From the conceptual management point of view, the database system has two hierarchy:

- ▶ Instance
- ▶ Database

When adding the two DB2 WLM hierarchy, superclasses and subclasses, there are total four levels of hierarchy. If AIX WLM, which has two level hierarchy, is going to be used to manage the database system from the instance level, you have to design carefully the mapping between DB2 instance, database, service classes, and AIX services classes.

You can choose from two different mapping schemes, flat mapping scheme and 1:1 mapping scheme.

Figure 8-6 illustrates flat mapping scheme. Flat mapping scheme is useful on servers that are used not only as the DB2 database server but also have other applications running. In this mixed environment, you would want to separate the application and the database activities from each other to have their own AIX WLM superclasses. All DB2 service classes would be mapped to their own subclasses under AIX WLM DB2 superclass.

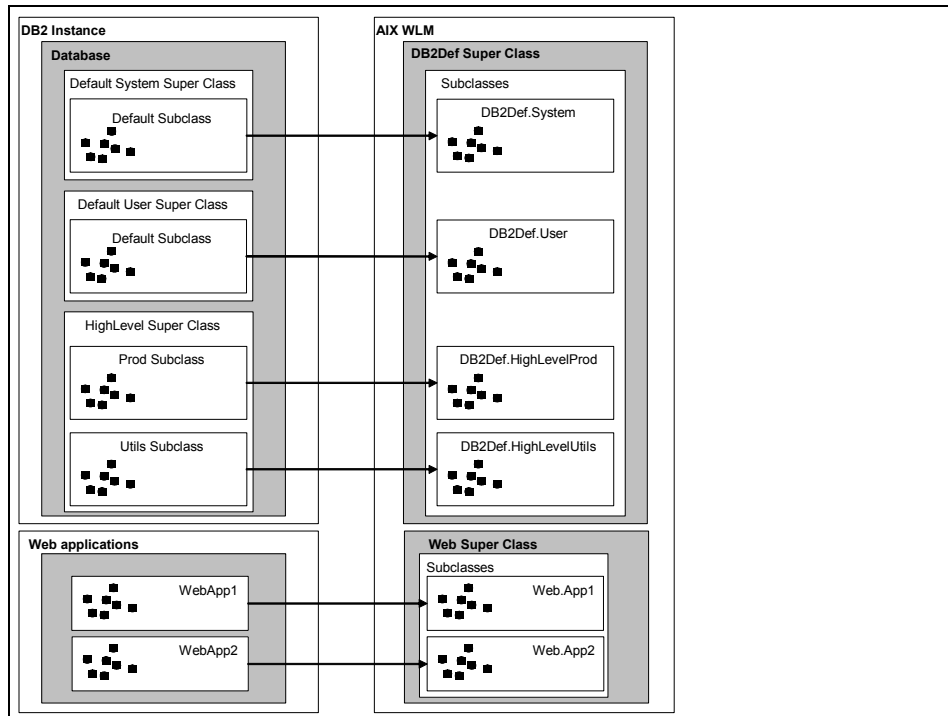


Figure 8-6 Flat mapping scheme

When you are running dedicated database server, we recommend you use 1:1 mapping scheme as illustrated in Figure 8-7.

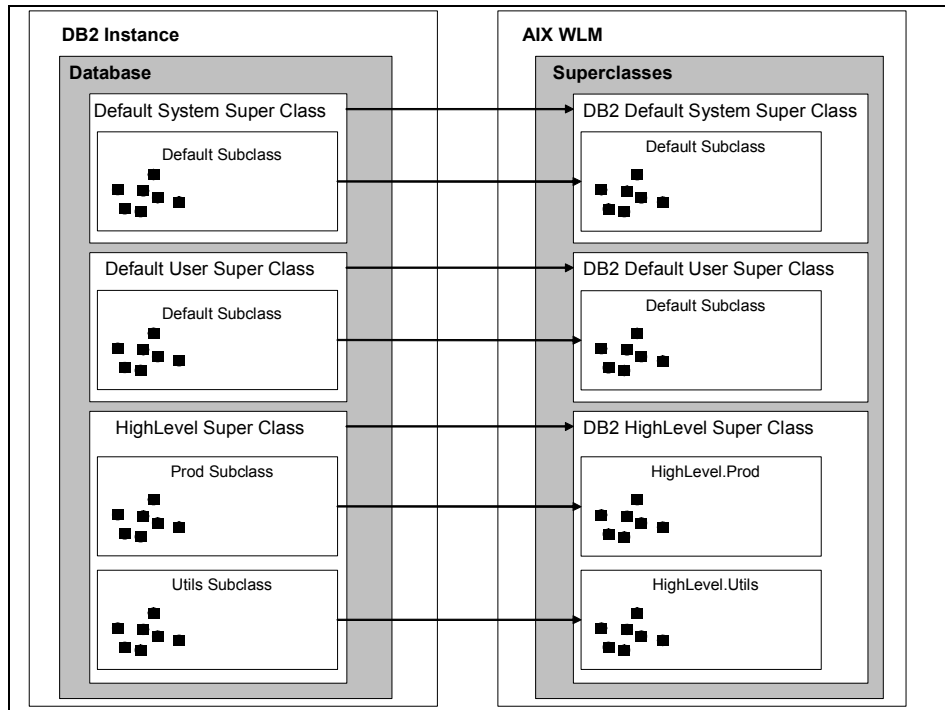


Figure 8-7 1:1 mapping

The main difference between flat and 1:1 mapping is that on 1:1 mapping, you can have identical super and classes on both DB2 WLM and AIX WLM. This makes implementing and maintenance much easier. We use 1:1 mapping scheme for our example in this chapter.

8.2.3 Integrating DB2 service classes with AIX service classes

When integrating DB2 WLM to AIX WLM, plan carefully and design properly are the keys to success. You should be able to maintain and monitor your environment easily. If you already have implemented DB2 WLM, you should examine if your DB2 WLM configuration can be integrated with AIX WLM. You also have to decide the mapping type: 1:1 mapping or flat mapping scheme.

Designing DB2 WLM and AIX WLM integration

Since we are running dedicated database server, we use 1:1 mapping as the mapping scheme. Figure 8-8 illustrates our lab DB2 WLM configuration. The database WLMDB has one user defined superclass HIGHLVL which has four subclasses. Four user workloads map those four subclasses.

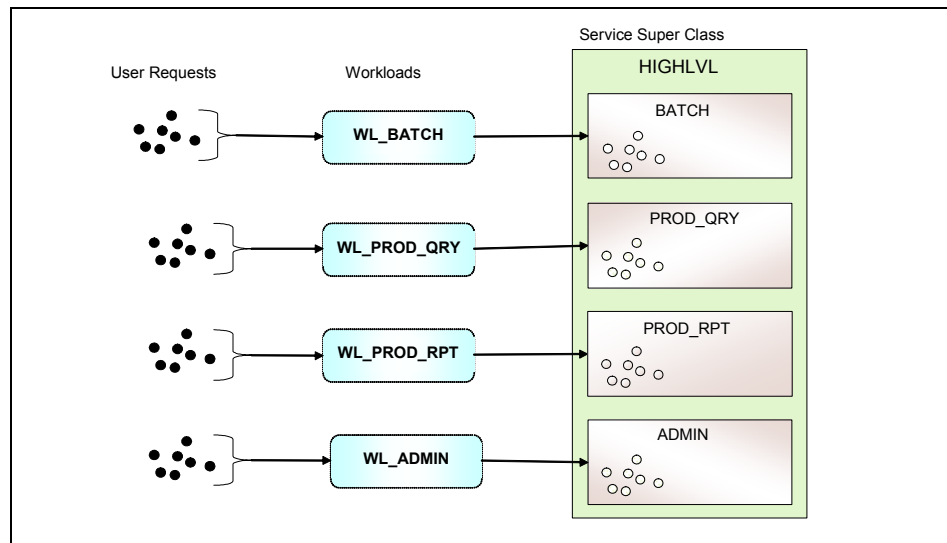


Figure 8-8 User workloads and service classes

Besides the user defined workloads and service classes, we have two default DB2 WLM system workloads and three default system service classes. All the system default service classes has one default subclass. So we have altogether four super service classes. We create equivalent service classes on AIX. The following lists DB2 service classes with equivalent AIX service classes:

- ▶ SYSDEFAULTSYSTEMCLASS → db2DefSys
- ▶ SYSDEFAULTUSERCLASS → db2DefUser
- ▶ SYSDEFAULTMAINTENANCECLASS → dbDefMnt
- ▶ HIGH_LVL → db2HighLevel
 - ADMINS → db2HighLevel.Admins
 - PROD_QRY → db2HighLevel.Prod_QRY
 - PROD_RPT → db2HighLevel.Prod_RPT
 - BATCH → db2HighLevel.Batch

On AIX WLM, resources will be allocated to service class depending on how much shares the service class has relatively to the total amount of shares. We have to decide how much CPU shares we are going to give for our service classes. Figure 8-9 shows the CPU shares allocation for our AIX WLM service classes. It also illustrates how the outbound correlators for each service class are defined. To be easily recognized, all our outbound correlators in this example start with _ (under score).

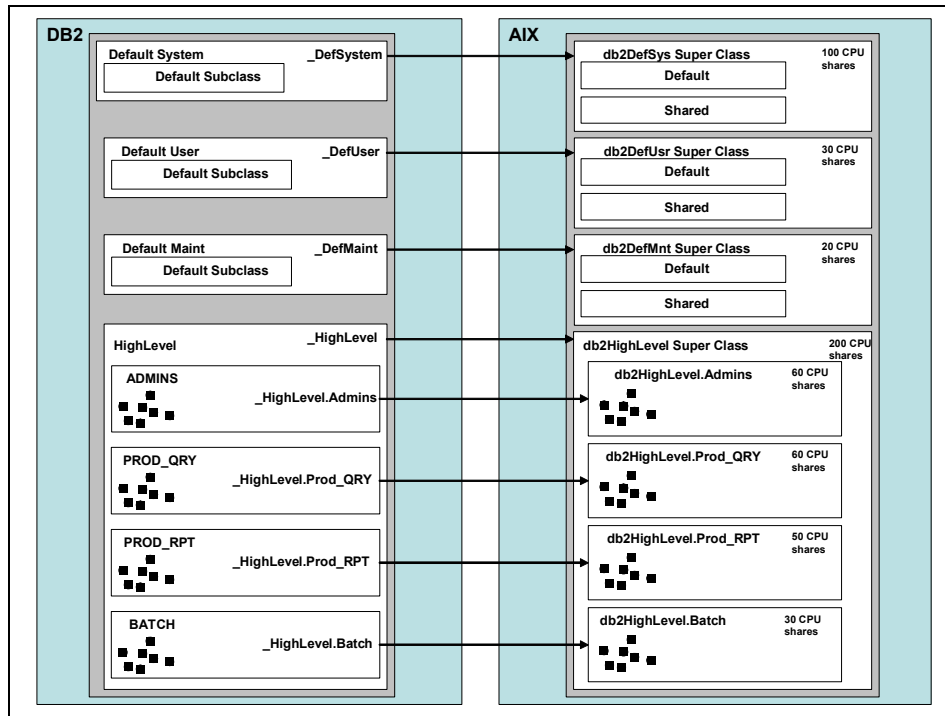


Figure 8-9 1:1 mapping from existing DB2 WLM configuration

Implementing DB2 WLM and AIX WLM integration

Now that we have designed our mapping scheme, we are ready to implement the mapping. We start from creating new configuration set, *production*, for AIX WLM. See Example 8-5.

Example 8-5 Creating new AIX WLM configuration set

```
# cd /etc/wlm
# cp -r template production
# wlmcntrl -ud production
```

Note that `wlmcntrl -ud` command requires that AIX WLM is already running. If AIX WLM is not running, use command `wlmcntrl -d product` instead.

Creating classes

After creating the configuration set, we can start configuring it. The first step is to create four superclasses and four subclasses under superclass `db2HighLevel` as shown in Example 8-6.

Example 8-6 Creating AIX WLM superclasses

```
# mkclass -a inheritance=no -c shares=100 db2DefSystem
# mkclass -a inheritance=no -c shares=20 db2DefMaint
# mkclass -a inheritance=no -c shares=30 db2DefUser
# mkclass -a inheritance=no -c shares=200 db2HighLevel
# mkclass -a inheritance=no -c shares=30 db2HighLevel.Batch
# mkclass -a inheritance=no -c shares=60 db2HighLevel.Prod_QRY
# mkclass -a inheritance=no -c shares=50 db2HighLevel.Prod_RPT
# mkclass -a inheritance=no -c shares=60 db2HighLevel.Admins
```

Creating mapping rules

Since the new AIX WLM configuration set *production* has been activated and running, we can create superclass mapping rules by editing the rules file in the directory `/etc/wlm/current`. Example 8-7 shows the contents of `/etc/wlm/current/rules`. Beware that entries in the rules file also imply the evaluation order.

Example 8-7 AIX WLM rules file

```
* IBM_PROLOG_BEGIN_TAG
* This is an automatically generated prolog.
*
* bos530 src/bos/etc/wlm/rules 1.2
*
* Licensed Materials - Property of IBM
*
* (C) COPYRIGHT International Business Machines Corp. 1999,2002
* All Rights Reserved
*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*
* IBM_PROLOG_END_TAG
* class resvd user group application type tag
System - root - - - -
db2DefSystem - - - - _DefSystem
db2DefMaint - - - - _DefMaint
db2DefUser - - - - _DefUser
db2HighLevel - - - - _HighLevel
db2HighLevel - - - - _HighLevel.Batch
db2HighLevel - - - - _HighLevel.Prod_QRY
db2HighLevel - - - - _HighLevel.Prod_RPT
db2HighLevel - - - - _HighLevel.Admins
Default - - - - - - -
```

Note: You should always leave the System class above the user defined classes, so system processes will go to their default service classes. You should also leave the Default user class below you user defined classes.

Next is to define the mapping rules for subclasses under db2HighLevel. The corresponding rules file for db2HighLevel is /etc/wlm/current/db2HighLevel/rules. We edit this file as shown in Example 8-8.

Example 8-8 Rules for subclasses under superclass db2HighLevel

```
* class resvd user group application type tag
Batch - - - - -_HighLevel.Batch
Prod_QRY - - - - -_HighLevel.Prod_QRY
Prod_RPT - - - - -_HighLevel.Prod_RPT
Admins - - - - -_HighLevel.Admins
```

Testing the classes and rules

Use `wlmcntr1-u` command to refresh the running AIX WLM configuration so that all the changes made will take effect. We check that all our newly defined service classes are in the system using `lsclass -fr` command as show in Example 8-9.

Example 8-9 List of service classes

```
# lsclass -fr
System:
    memorymin = 1

Default:

Shared:

db2DefSystem:
    inheritance = "no"
    CPUshares = 100

db2DefMaint:
    inheritance = "no"
    CPUshares = 20

db2DefUser:
    inheritance = "no"
    CPUshares = 30

db2HighLevel:
    inheritance = "no"
```

```

CPUshares = 200

db2HighLevel.Default:

db2HighLevel.Shared:

db2HighLevel.Batch:
    inheritance = "no"
    CPUshares = 30

db2HighLevel.Prod_QRY:
    inheritance = "no"
    CPUshares = 60

db2HighLevel.Prod_RPT:
    inheritance = "no"
    CPUshares = 50

db2HighLevel.Admins:
    inheritance = "no"
    CPUshares = 60

```

We can test our configuration by checking to which AIX WLM service class the processes and threads with AIX WLM tag `_HighLevel.Batch` are assigned, You can use `wlmcheck` command as shown in Example 8-10.

Example 8-10 Test mapping rules

```

# wlmcheck -a "- - - - _HighLevel.Batch"
System
db2HighLevel.Batch

```

As we can see from Example 8-10, all processes or threads which are tagged with AIX WLM tag “`_HighLevel.Batch`” will be mapped to service subclass `db2HighLevel.Batch`, unless the process or thread is owned by *root*. If process or thread is owned by *root*, it will be mapped to system default service class. In our case, all DB2 processes and threads are owned by instance owner *db2inst1*, which ensures that our mappings will work as expected.

If your database is partitioned, you must set up AIX WLM on all nodes. To setup another node, you only need to copy, by ftp or scp, the AIX WLM configuration directory to that node and activate it. Example 8-11 shows how we set up AIX WLM on the second physical database node Bonnie in our Lab environment.

Example 8-11 Copying AIX WLM configuration from node to another

```

> scp -r clyde:/etc/wlm/production /etc/wlm/

```

```
...  
> wlmcntrl -d production
```

Note that `wlmcntrl` command is executed without `-u` flag since the AIX WLM was not running while the command is executed.

Setting up tags for DB2 service classes

Once the AIX WLM environment is ready and running, the last step is to “link” DB2 service classes with AIX WLM service classes. This is accomplished by alter the DB2 service classes outbound correlator to associate the AIX WLM tags with them.

For the default DB2 service classes, the only modification allowed is to set the outbound correlator. This is for mapping the DB2 default service classes to AIX WLM service classes.

Similar to DB2 WLM, every AIX WLM superclass has at least one subclass for processes. On AIX WLM, there are two default subclasses:

- ▶ Default: for all processes which are not mapped to any other subclass.
- ▶ Shared: All memory pages which contain processes belonging to more than one sub class under same superclass will be assigned to this subclass.

When the outbound correlator for the default DB2 WLM superclasses is set, the default subclass inherits the setting and is mapped to the correct superclass on AIX WLM. The workload management works will be done under default AIX WLM subclass, which inherits all its attributes from parent superclass. This is why we do not need to define rules for default AIX WLM subclasses, nor we do not have to create additional subclasses for default AIX WLM superclasses.

Example 8-12 and Example 8-13 show how to set up outbound correlator for default and user defined DB2 WLM service classes.

Example 8-12 Set outbound correlators for default service classes

```
ALTER SERVICE CLASS sysdefaultsystemclass OUTBOUND CORRELATOR _DefSystem;  
ALTER SERVICE CLASS sysdefaultmaintenanceclass OUTBOUND CORRELATOR _DefMaint;  
ALTER SERVICE CLASS sysdefaultuserclass OUTBOUND CORRELATOR _DefUser;
```

In Example 8-13, we set outbound correlator for our HIGHLVL service superclass and subclasses running under it.

Example 8-13 Set outbound correlators for user superclass and subclasses

```
ALTER SERVICE CLASS highlvl OUTBOUND CORRELATOR _HighLevel;  
ALTER SERVICE CLASS prod_qry UNDER HIGHLVL OUTBOUND CORRELATOR  
_HighLevel.Admins;
```

```
ALTER SERVICE CLASS batch UNDER highlvl OUTBOUND CORRELATOR _HighLevel.Batch;
ALTER SERVICE CLASS prod_qry UNDER highlvl OUTBOUND CORRELATOR
_HighLevel.Prod_QRY;
ALTER SERVICE CLASS prod_rpt UNDER highlvl OUTBOUND CORRELATOR
_HighLevel.Prod_RPT;
```

You can check that the service class outbound correlators are set up properly by looking the system catalog table SYSCAT.SERVICECLASSES as show in Example 8-14.

Example 8-14 Check service class outbound correlator setting

```
db2 "select substr(char(SERVICECLASSID),1,2) as ID,
substr(SERVICECLASSNAME,1,19) as SERVICECLASSNAME,
substr(PARENTSERVICECLASSNAME,1,26) as PARENTSERVICECLASSNAME,
substr(OUTBOUNDCORRELATOR,1,19) as TAG from syscat.serviceclasses"
```

ID	SERVICECLASSNAME	PARENTSERVICECLASSNAME	TAG
1	SYSDEFAULTSYSTEMCLASS	-	_DefSystem
2	SYSDEFAULTMAINTENAN	-	_DefMaint
3	SYSDEFAULTUSERCLASS	-	_DefUser
11	SYSDEFAULTSUBCLASS	SYSDEFAULTSYSTEMCLASS	-
12	SYSDEFAULTSUBCLASS	SYSDEFAULTMAINTENANCECLASS	-
13	SYSDEFAULTSUBCLASS	SYSDEFAULTUSERCLASS	-
14	HIGHLVL	-	_HighLevel
15	SYSDEFAULTSUBCLASS	HIGHLVL	-
16	ADMINS	HIGHLVL	_HighLevel.Admins
17	BATCH	HIGHLVL	_HighLevel.Batch
18	PROD_RPT	HIGHLVL	_HighLevel.Prod_RPT
19	PROD_QRY	HIGHLVL	_HighLevel.Prod_QRY

12 record(s) selected.

Verifying OS level mapping

In addition to verify that the application tags are set properly. We want to be sure that the actual OS level mapping really happens. We can check if the default DB2 system service class is getting mapped to correct AIX WLM service class using **ps** command as show in Example 8-15.

Example 8-15 Check mapping is working for default system service class

```
>ps -ef | egrep "db2sysc|PID"
      UID      PID      PPID      C    STIME    TTY    TIME CMD
db2inst1 942242 1437962    0 15:48:10    -    1:24 db2sysc 1
db2inst1 1347754 1839122    0 15:48:11    -    0:09 db2sysc 2
db2inst1 2912646 1528048    2 15:48:10    -    1:07 db2sysc 0
```

```
>ps -m -o THREAD,class -p 2912646 | egrep "db2DefSystem|PID"
...      TID ST  CP PRI SC ...      F ... CLASS
... 1249349 S    0 60 1 ... 400400 ... db2DefSystem
... 3297525 S    0 60 1 ... 400400 ... db2DefSystem
... 3600475 S    0 60 1 ... 400400 ... db2DefSystem
... 3727573 S    0 60 1 ... 400400 ... db2DefSystem
... 700755 S    0 60 1 ... 400400 ... db2DefSystem
... 794987 S    0 60 1 ... 400400 ... db2DefSystem
... 1405319 S    0 60 1 ... 400400 ... db2DefSystem
... 1782027 S    0 60 1 ... 400400 ... db2DefSystem
... 4051391 S    0 60 1 ... 400400 ... db2DefSystem
```

The first `ps` command is to obtain the process ID for `db2sysc` process on `partition0`. Then we used this process ID to find out if the DB2 agents are mapped to AIX WLM service classes. This is a easy way to verify if the mapping is working correctly. You can use the same approach to check if the default maintenance service class is mapped as defined. Before checking if the user defined DB2 workloads are mapped to correct AIX service classes, check first if the workloads are mapped to correct DB2 service classes.

Now we have set up four user super service classes for AIX WLM. Three of them are mapped to default DB2 WLM superclasses and one for DB2 WLM user superclass. Figure 8-10 illustrates how CPU resources are shared by AIX WLM service superclasses.

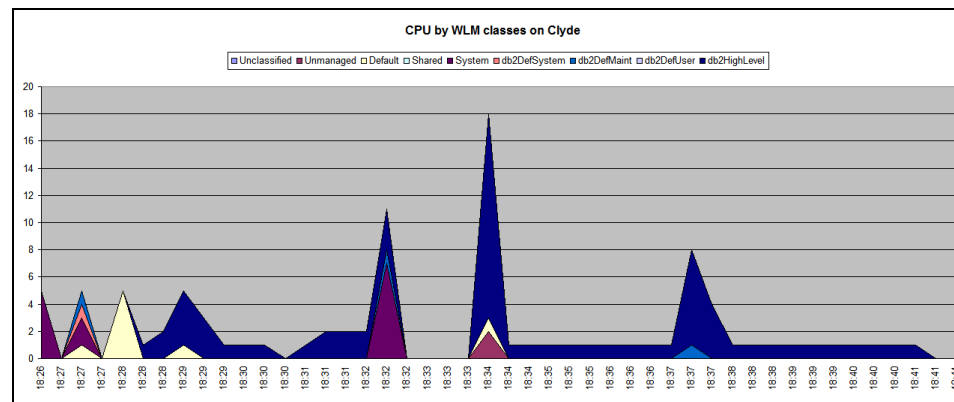


Figure 8-10 CPU by classes

The chart shows that our dedicated database server is currently running quite light load. There seems to be almost no CPU activity for `db2DefSystem` and `db2Defmaint`. These classes are 1:1 mapped with DB2 service classes `SYSDEFAULTSYSTEMCLASS` and `SYSDEFAULTMAINTENANCECLASS`.

There is no activity for db2DefUser at all since currently no workloads are mapped to DB2 WLM service class SYSDEFAULTUSERCLASS. db2HighLevel has some activity as all DB2 user workloads are running under this service superclass. From the chart you can also see that there is a gap between 18:32 and 18:33. Since the chart only shows CPU activity, we should also examine memory and disk usage to find out what has caused the gap.

8.2.4 Monitoring

AIX offers excellent monitoring tools and some of them are AIX WLM aware. These tools provides us a fast way to monitor how the workloads are running on their service classes. This gives us the capability to monitor the databases with standard operating system monitoring tools much more effectively than it have never been before.

Basic AIX WLM monitoring tools

In this section, we concentrate on the following AIX WLM aware monitoring tools:

- ▶ topas
- ▶ ps
- ▶ wlmstat
- ▶ nmon

topas, *ps*, and *wlmstat* comes with standard AIX operating system. *nmon* is freely downloadable from

<http://www.ibm.com/collaboration/wiki/display/WikiPtype/nmon>

topas

Topas is a usefull and user friendly tool for monitoring your system real time. Topas provides a good view of overall system performance and you can include AIX WLM data to it. You can specify how many top AIX WLM service classes you want to include in topas output by specifying the number of WLM classes with **-w** flag.

Figure 8-11 illustrates the output of **topas -w 10**.

```

X topas
Topas Monitor for host:  clyde          EVENTS/QUEUES  FILE/TTY
Thu Sep 12 01:13:20 2007  Interval:  2
Kernel  6.9  I##                               Reads         39  Rawin      0
User    1.3  I#                                Writes         18  Ttyout    373
Wait    7.0  I###                               Forks          0  Igets     0
Idle    84.7 I#####                               Execs          0  Namei     67
Physc = 0.06                               %Entc= 11.9  Runqueue    0.0  Dirblk    0
                                           Waitqueue    0.5

Network KBPS  I-Pack  0-Pack  KB-In  KB-Out
lo0      19.8   40.4   40.4   9.9    9.9
en0      3.7    5.5    0.0    3.7    0.0
en1      1.8    5.5    5.0    0.7    1.1

Disk  Busy%  KBPS  TPS  KB-Read  KB-Writ
hdisk1 15.4   14.0  3.5  14.0    0.0
hdisk3 0.0    12.0  1.0  0.0    12.0
hdisk2 0.0    0.0   0.0  0.0    0.0
hdisk4 0.0    0.0   0.0  0.0    0.0
hdisk5 0.0    0.0   0.0  0.0    0.0
hdisk6 0.0    0.0   0.0  0.0    0.0
hdisk7 0.0    0.0   0.0  0.0    0.0
hdisk0 0.0    0.0   0.0  0.0    0.0

PAGING
Faults  37  Real,MB  8192
Steals  0  % Comp   71.8
PgspIn  3  % Noncomp 28.1
PgspOut 0  % Client  24.5

PageIn  3
PageOut 2
Sios    6  PAGING SPACE
        Size,MB  4096
        % Used   34.3
NFS (calls/sec)
ServerV2 0
ClientV2 0  Press:
ServerV3 0  "h" for help
ClientV3 0  "q" to quit

WLM-Class (Active)  CPU%  Mem%  Disk-I/O%
System              5     33    0
db2DefSystem        1     2     0
db2DefMaint         0     0     0
db2DefUser          0     0     0
db2HighLevel        9    12    0
Default             2    53    1
Unmanaged           1    26    0
Shared              0    13    0
Unclassified        0     0     0

```

Figure 8-11 Using topas for overall system monitoring

To view only real time CPU usage between service classes, use **topas -W**. Example 8-16 shows how the output may look like.

Example 8-16 Using topas for WLM monitoring

```

Topas Monitor for host:  clyde          Interval:  2   Tue Sep 11 22:37:29
2007
WLM-Class (Active)      CPU%    Mem%    Disk-I/O%
System                  0       5       0
db2DefSystem            1       2       0
db2DefMaint             0       0       0
db2DefUser              0       0       0
db2HighLevel            9      12       0
Unmanaged               8       4       0
Default                 1       5       0
Shared                  0       2       0
Unclassified            0       4       0

```

ps

ps command is very useful to get information about processes and threads. In Example 8-14 on page 212, we show how to find the mappings between DB2 service classes and AIX WLM service classes from DB2 side using **ps** command. For more details about **ps** command, refer to AIX manual pages.

wlmstat

If you are familiar with **iostat** and **vmstat**, you have no difficulties using **wlmstat** command. It displays all the super and service classes with current CPU, memory, and disk I/O usage. See Example 8-17. **wlmstat** accepts two numeric input parameters. The first one is interval and the second one is number of cycles. For more detailed usage information for **wlmstat** see **wlmstat** manual pages.

Example 8-17 Using wlmstat for WLM monitoring

```
>wlmstat
```

CLASS	CPU	MEM	DKIO
Unclassified	0	0	0
Unmanaged	0	16	0
Default	0	18	0
Shared	0	11	0
System	0	5	0
db2DefSystem	0	0	0
db2DefMaint	0	0	0
db2DefUser	0	0	0
db2HighLevel	0	0	0
db2HighLevel.Default	-	-	-
db2HighLevel.Shared	-	-	-
db2HighLevel.Batch	-	-	-
db2HighLevel.Prod_QRY	-	-	-
db2HighLevel.Prod_RPT	-	-	-
db2HighLevel.Admins	-	-	-
TOTAL	0	34	0

nmon

nmon is a free tool for analyzing AIX performance. It is not part of standard AIX operating environment and is not officially supported by IBM. **nmon** has user friendly environment, which suites very well for real time monitoring. Using **nmon**, you can see your CPU, disk I/O, and memory usage real time one screen. The rich monitoring capability has made **nmon** a very popular tool for monitoring AIX and its resources. Because it has WLM enhancements, we are able to use **nmon** to monitoring AIX WLM.

Now that we have integrated the DB2 WLM with AIX WLM, we can see how the DB2 workloads are behaving and compare the results with the overall operating

system performance. CPU utilization, workload manager activities, and memory usage all can be seen realtime in one screen, which is very useful in finding bottlenecks and solving performance problems on the database system.

Figure 8-12 on page 218 presents a simple realtime monitoring for WLM super and subclasses. With a single glance, we see that currently our four CPUs are utilized with 42.4% for user processes, 1.4% for system processes, and 53.9% for I/O wait. We are using 6345.8MB physical memory and have 1846.2MB free physical memory. At the time of observation, all our workloads were executed on DB2 service subclass PROD_RPT under HIGHLVL, which is mapped to AIX WLM service subclass db2HighLevel.Prod_RPT. We might be interested to examine this data further, since we probably want to know why and what our CPUs are waiting.

You can also take a look at AIX WLM configuration from `nmon` real time data. We are able to find CPU shares and tiers for different DB2 service classes. By having all this information on one central place, we are able to see how our overall system is performing at a glance.

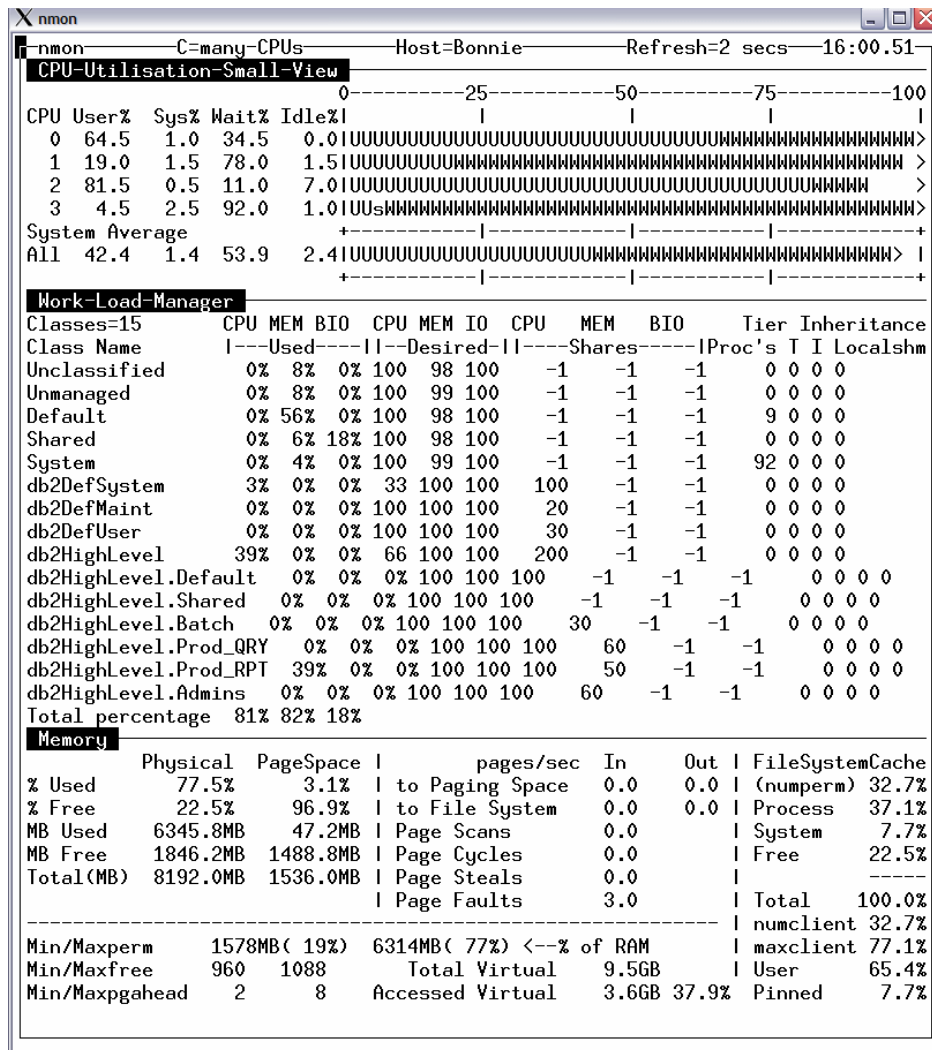


Figure 8-12 Real-time monitoring service classes with nmon

nmon is also capable of saving monitored data to file in comma separated values (CSV) format. It is an ideal tool to collect data for analysis using charts. Many of our AIX WLM related charts in this chapter were created using nmon.

In Figure 8-10 on page 213, we presented CPU usage chart by service classes. We noticed that there was a gap on CPU usage between 6:32 PM and 6:33 PM. The data was collected with nmon and saved as a CSV format file. The data are then imported to a spreadsheet for plotting charts. When data is collected using

operating system tools, not only the DB2 related data, but the entire operating system related data is collected.

Monitoring system using AIX WLM and DB2 WLM monitoring

In Figure 8-8 on page 206, we present a CPU usage by service classes chart using data from nmon. We notice that there was a gap on CPU usage between 6:32 PM and 6:33 PM and would like to know the cause of it. Since the data was collected with an operating system tool, the chart reflects the activities on the entire system including DB2 and non-DB2. To investigate the root cause, we first examine the average DB2 query execution time during that time frame. We collect the statistics using event monitor to table SCSTATS_BASIC_MON and plot the query average execution time between 6:22 PM and 6:42 PM as shown in Figure 8-13.

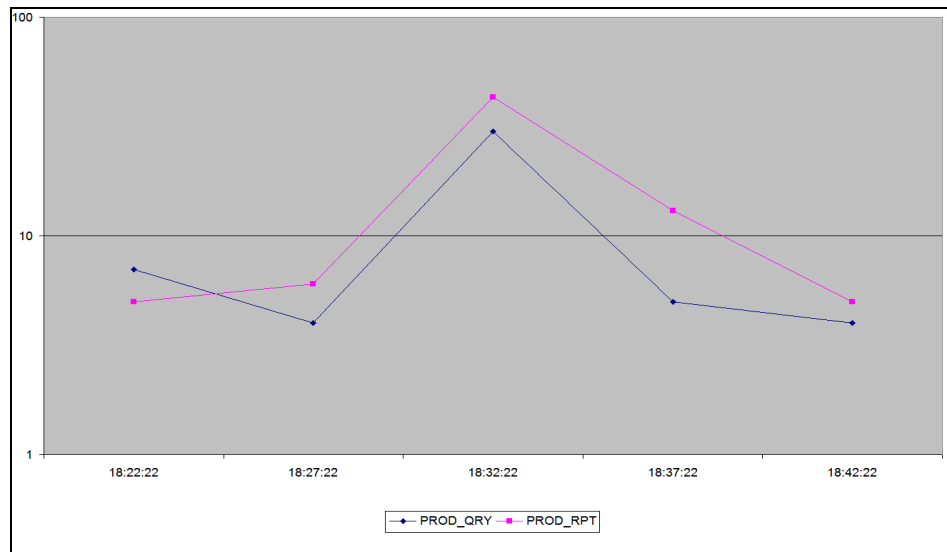


Figure 8-13 Average execution time

The CPU gap could have happened if there were no queries at the time, but this was not the case. We see that there was a peak for average execution time on both PROD_RPT and PROD_QRY service classes at 6:32 PM. This peak can imply lack of the operating system resources.

We then examine the operating system disk I/O activities at the time of this incidence using the same data file collected by nmon. Figure 8-14 show the disk reads between 6:26 PM and 6:40 PM.

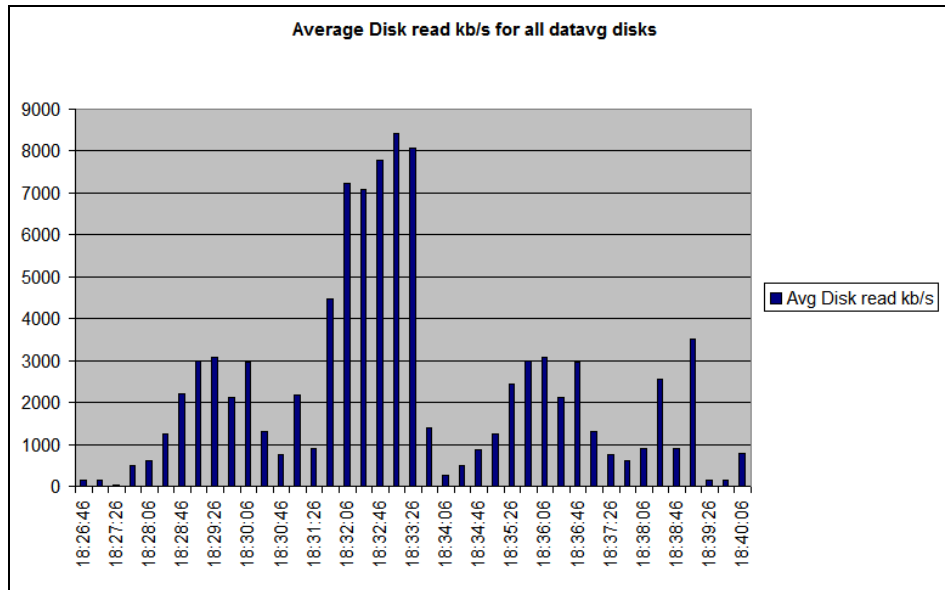


Figure 8-14 Operating system disk reads

The disk reads in the chart are average for all the disks which belongs to datavg volume group. This volume group holds all our database related data. There was a high disk reads between 6:32 PM and 6:33 PM due to the operating system administrator mistakenly started the backup procedure for the volume group and corrected this action quickly. DB2 queries was in I/O wait and did not consume CPU time.

We were able to find the root cause for the relatively short service break down by combining the usage of DB2 WLM and AIX WLM monitoring tools. By using AIX WLM you are also able to prioritize your database related tasks over other tasks to provide the optimal database performance.



WLM sample scenarios - other usage

In this chapter, we demonstrate how to use DB2 WLM beyond just managing DB2 work. Using the same data captured in the previous scenarios, WLM shows its usefulness in solving other tasks in running a data center. We explore two sample scenarios:

- ▶ Capacity planning- How to use WLM to trend and project resource consumption and needs.
- ▶ Chargeback accounting- How to use WLM to establish a method of capturing resource utilization for the purposes of charging for resource usage.

9.1 Capacity planning

Capacity planning has two main objectives:

- ▶ Trending - Establish resource usage and workload trends over an extended period of time.
- ▶ Projecting - Once trends have been established, project future resource requirements based on historical data capture and analysis.

Effective capacity planning can be very complex in large complex environments and involve trending and projecting many resources. For simplicity, our example covers a single resource, CPU. However, the principles can be applied to other resources such as memory.

We want to know, how much CPU resources are we using today so we can project if we have adequate resources tomorrow when the workloads change. Such projections are needed when business grows and more data is added to our database; new applications are added and the number of workloads will increase; or more users are expected to use the database and the number of queries will increase. These and other questions can be answered by capturing WLM monitoring data.

To perform capacity planning data trending in our sample scenario, the DB2 WLM monitoring data needed for specific time periods is:

- ▶ Number of concurrent active connections
- ▶ Total request execution time
- ▶ Average request execution time

This information is contained in our event monitor table SCSTATS_BASIC_MON. Using these columns along with AIX CPU usage data provided by either `nmon` or `vmstat`, we can correlate the two monitoring elements to provide a simple trending chart. In our example, we collect the CPU data from each server using `nmon` which outputs the needed data using the following command:

```
nmon -f -s1800 -c48 -rw1m
```

This creates an output file that can be input to the `nmon analyser.xls`. Both `nmon` and the `nmon analyser.xls` are available for download from the following IBM Wiki Web site:

<http://www.ibm.com/collaboration/wiki/display/WikiPtype/nmon>

9.1.1 The workload environment

We start our capacity planning scenario by using the environment we established in Chapter 4. Here we are interested in trending and projecting our production workloads for production queries WL_PROD_QRY and production reports WL_PROD_RPT as shown in Figure 9-1. The other workloads are of no immediate concern and are ignored in this discussion.

The followings are the monitoring requirements for capacity planning:

- ▶ The retention period of the monitoring data is 12 months.
- ▶ The monitoring Intervals is 20 minutes.

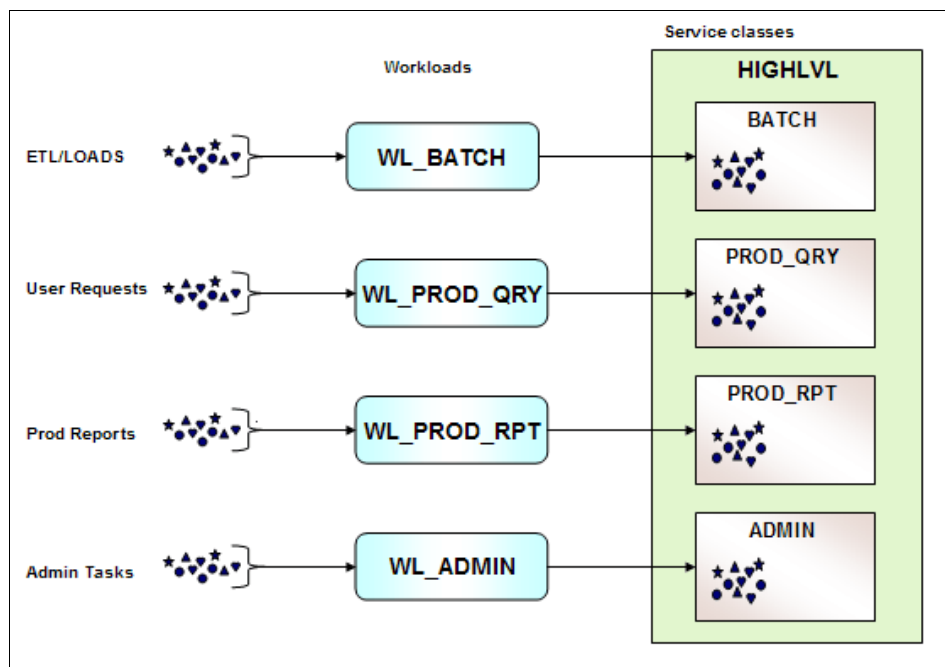


Figure 9-1 Workload environment

9.1.2 Collecting the trending data

During running sample workloads, the statistics information is captured and sent to the statistics event monitor at 30 minutes interval automatically. We use the SCSTATS_BASIC_MON table to monitor the base information for capacity planning.

9.1.3 Monitoring and analysis

We used the query shown in Example 9-1 to collect the data for this scenario.

Example 9-1 Query to capture capacity planning data

```

SELECT
    statistics_date,
    statistics_hour,
    subclass_name,
    con_act_top,
    avg_r_exe_tm,
    DECIMAL(avg_r_exe_tm/con_act_top,9,3) AS avg_r_exe_tm_per_act
FROM (SELECT
    DATE(statistics_timestamp) AS statistics_date,
    HOUR(statistics_timestamp) AS statistics_hour,
    SUBSTR(service_subclass_name,1,15) AS subclass_name,
    CASE WHEN 1 > INT(SUM(concurrent_act_top))
        THEN 1
        ELSE INT(SUM(concurrent_act_top))
    END AS con_act_top,
    CASE WHEN 1 > INT(SUM(request_exec_time_avg))
        THEN 1
        ELSE
            ECIMAL(SUM(request_exec_time_avg) / 1000,9,3)
        END AS avg_r_exe_tm
    FROM scstats_basic_mon
    WHERE service_subclass_name in ('PROD_RPT', 'PROD_QRY')
    GROUP BY DATE(statistics_timestamp),
             HOUR(statistics_timestamp),
             service_subclass_name
    ORDER BY DATE(statistics_timestamp),
             HOUR(statistics_timestamp),
             service_subclass_name);

```

Now that we have collected the data, we can create various charts for analysis. For this example, we have collected three weeks of data, covering the primetime hours. Our first chart shown in Figure 9-2 is the total request execution times across all data partitions for both PROD_QRY and PROD_RPT service classes. For ease of viewing, only the primetime hours are shown as this is our main area of focus. From this chart we can see which hours has the highest request execution times and if a trend appears to be established. Here we clearly see that the 11 o'clock hour has the highest total request execution times. Additionally, we appear to have established an upward trend in request execution

times. Notice that the request execution times drops off dramatically after 5:00 PM and continues to remain low for the rest of the primetime shift.

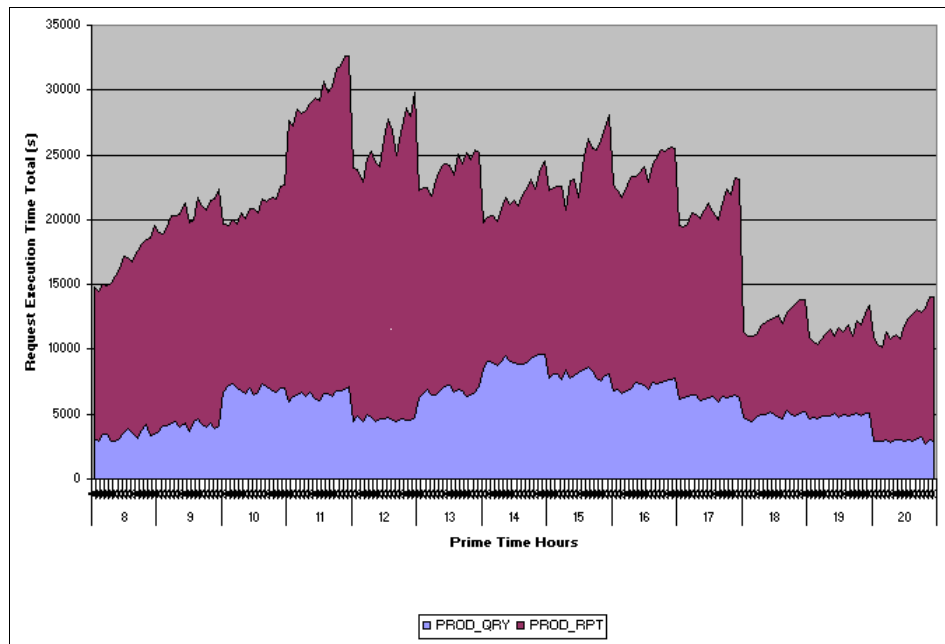


Figure 9-2 Total Request Execution times in seconds, across all partitions

Next we compare the CPU utilization across the same time periods to give us a perspective of how our total request executions times related to CPU consumption. In Figure 9-3 on page 226, we see the total CPU percentage for all our partitions. This can easily be captured using `vmstat` or `nmon`.

Now we can put into perspective this chart. We see that our peak CPU consumption is at 11 AM. We top out at 90% when the total request execution time is 32,635 seconds. So we can make a basic assumption that we maximum out at 100% when we hit total request execution time of 36,261 ($32,635 / .90$). Other factors play into CPU consumption but for simple planning purpose, this gives us a simple model.

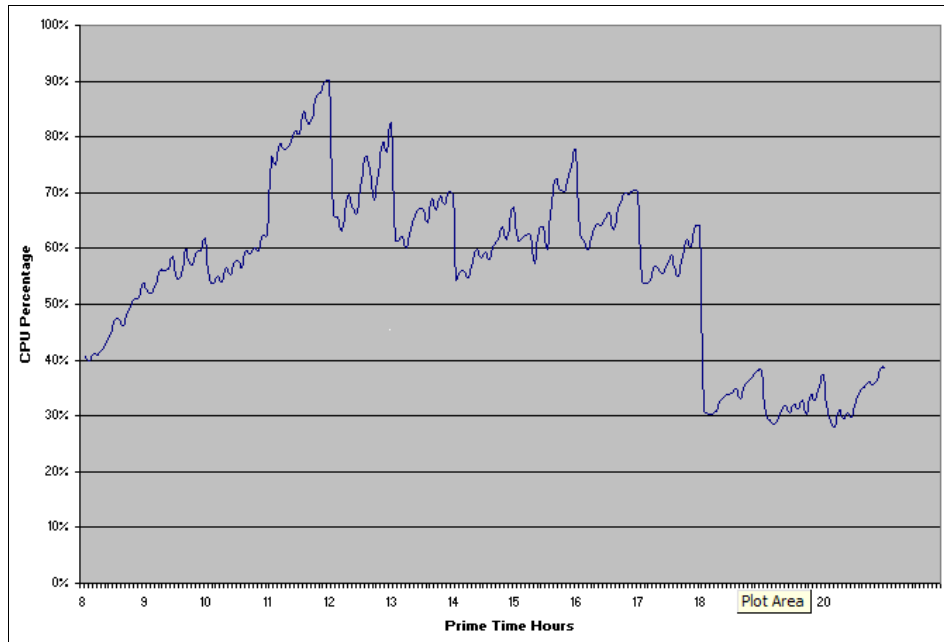


Figure 9-3 Total CPU percentages across all partitions

Next we need to analyze if any correlation exists between request execution times and the number of active workloads. In Figure 9-4 on page 227, we have a chart showing the number of active workloads activities for the same time periods.

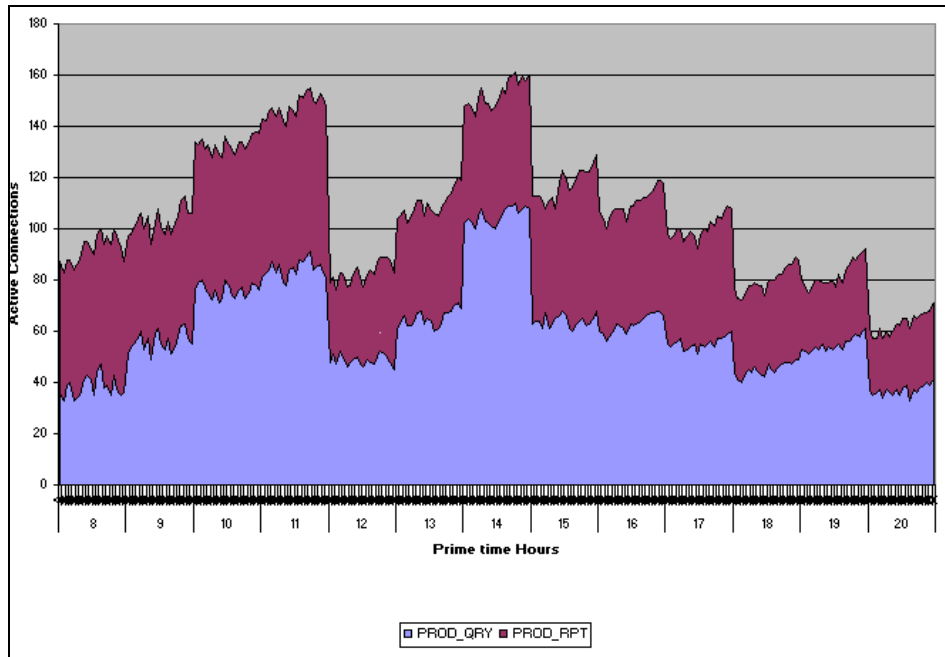


Figure 9-4 Active workload activities

Here we see a slightly different picture. we have two periods of high workloads, 11AM and 2PM. The difference must be in the average request execution times for each workload activity. In Figure 9-5 on page 228, we have a chart showing the average request execution time for average active workloads.

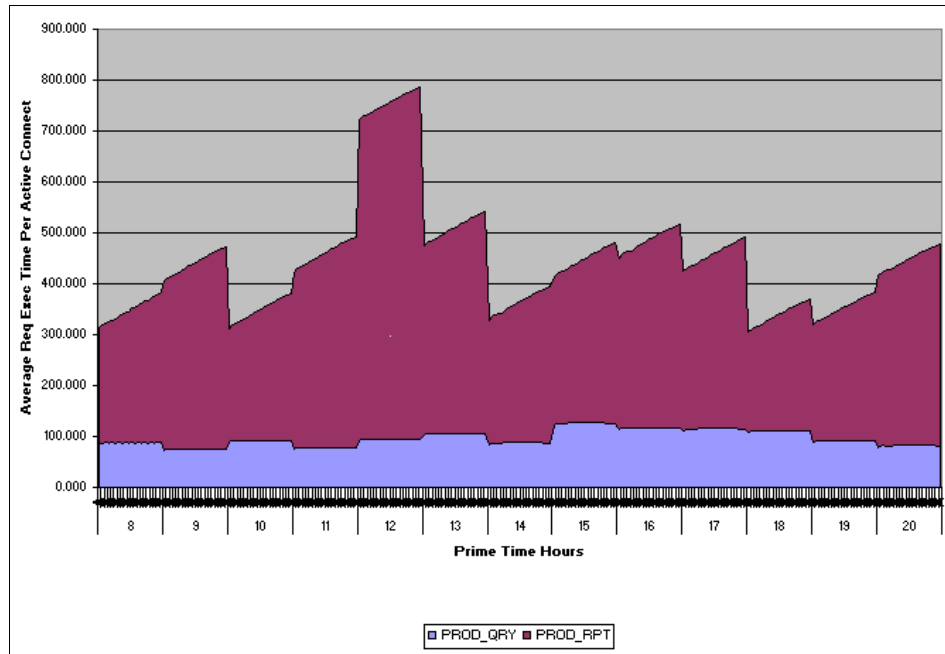


Figure 9-5 Average request execution times per active connection

What a surprise, our most “expensive” requests are submitted during lunch time (12PM). We can also confirm that the workloads at 2PM are slightly more CPU intensive than those submitted at 11AM.

All of our charts confirm that a linear co-relationship exists between request execution times and percentage of CPU. As the total request execution time goes up so does the total percentage of CPU.

9.1.4 Summary

From our simple capacity planning example, we can see that WLM provides an effective means for doing capacity planning. We must note that several assumptions were made in doing the analysis:

- ▶ The linear correlation between the WLM request execution times and CPU utilization exists. When the WLM request execution times go up so does the CPU utilization. If this relationship is not linear, more complex mathematical analysis must be used to establish the proper relationship.
- ▶ A linear correlation exists between the WLM number of concurrent active connections and the WLM request execution times. Otherwise, the averages will be misleading.

These two important relationships should be monitored and validated periodically.

Our example scenario shows us several important facts:

- ▶ Our workload activity is not evenly distributed across our primetime shift.
- ▶ 11AM is our highest CPU demand period due to the high number of requests.
- ▶ 12PM is a time when our most CPU intensive workloads are submitted. The workloads in this time period may need to be investigated to see if the CPU consumption of these workloads can be reduced or moved to a later time period.
- ▶ 2 PM is another time period that appears to be growing consistently and may need closer monitoring.
- ▶ Our trending line shows these time periods appear to be growing in total CPU consumption.
- ▶ If we can redistribute work from the middle of our primetime shift to later in the shift, we can possibly forestall needing to expand our capacity.

9.2 Chargeback accounting

In many customer environments, a single DB2 server may be shared and running several instances of DB2. Occasionally, a single instance may have multiple databases. In order to adequately charge end users for their resource consumption, chargeback accounting is needed. Several techniques are explored in this section to accomplish chargeback accounting. The example used in this sections shows two instances of DB2 with each instance containing a database. We want each instance/database charged to their respective departments.

Chargeback information by WLM

Depending on the technique used, the following data is needed for chargeback accounting

- ▶ DB2 WLM - Total request execution time, from Aggregate Request Data
- ▶ AIX WLM - percentage of CPU utilization for each instance

This scenario covers how to use DB2 WLM only. We covered the AIX WLM in Chapter 10, “DB2 WLM and DWE Design Studio” on page 235.

9.2.1 Business objectives

In our environment, we have two departments (Sales and Accounting) sharing the same server but running in separate instances.

9.2.2 Defining workload profile

Example 9-2 shows the workload management configuration for chargeback accounting. Each instance is configured using the same blueprint. For chargeback accounting information, only Request Execution times are needed.

Example 9-2 WLM configuration for chargeback accounting

```
--
-- instance db2inst01
--
CREATE SERVICE CLASS sales COLLECT AGGREGATE REQUEST DATA BASE DISABLE;
ALTER SERVICE CLASS sales ENABLE;

CREATE EVENT MONITOR basic_mon FOR STATISTICS WRITE TO TABLE
  SCSTATS (TABLE SCSTATS_BASIC_MON IN MAINT) AUTOSTART;
SET EVENT MONITOR basic_mon STATE 1;
--
-- instance db2inst02
--
CREATE SERVICE CLASS sales COLLECT AGGREGATE REQUEST DATA BASE DISABLE;
ALTER SERVICE CLASS sales ENABLE;

CREATE EVENT MONITOR basic_mon FOR STATISTICS WRITE TO TABLE
  SCSTATS (TABLE SCSTATS_BASIC_MON IN MAINT) AUTOSTART;
SET EVENT MONITOR BASIC_MON STATE 1;
```

9.2.3 Monitoring

After we have run our WLM setup for a week, we can analyze our data. Using the query in Example 9-3, we can capture the chargeback data from each instance. Since the event monitor names are the same but the superclass names are unique, we simply run the query in both instances.

Example 9-3 Query to collect chargeback accounting data

```
SELECT
  statistics_date,
  superclass_name,
```

```

DECIMAL(avg_r_exe_tm/con_act_top,9,3) AS avg_r_exe_tm_per_act
FROM (SELECT
  DATE(statistics_timestamp) AS statistics_date,
  SUBSTR(service_superclass_name,1,15) AS superclass_name,
  CASE WHEN 1 > DECIMAL(SUM(request_exec_time_avg),9,3)
    THEN 0
    ELSE DECIMAL(SUM(request_exec_time_avg) / 1000,9,3)
  END AS avg_r_exe_tm
FROM scstats_basic_mon
GROUP BY DATE(statistics_timestamp),
  service_superclass_name)
ORDER BY DATE(statistics_timestamp),
  service_superclass_name);

```

From this data we can create a spreadsheet as shown in Figure 9-6. Several points need to be mentioned here. Note that the total times vary from day to day but the percentage calculations are based on the total for the particular day. This way each department is charged for their percentage of use that day and the sum of the percentages is always 100%. If for example SALES was the only instance running that day, they would be charged 100%. This eliminates having a shortfall or gap in accounting for the machine usage.

STATISTICS_DATE	Total Request Execution Time(S)			Percentage Chargeback	
	ACCOUNTING	SALES	Grand Total	ACCOUNTING	SALES
08/27/07	98,940.58	218,744.34	317,684.93	31.144	68.856
08/28/07	57,731.21	221,394.80	279,126.02	20.683	79.317
08/29/07	80,126.25	229,726.85	309,853.11	25.859	74.141
08/30/07	111,796.64	224,795.77	336,592.41	33.214	66.786
08/31/07	89,442.65	235,012.39	324,455.05	27.567	72.433

Figure 9-6 Chargebase accounting spreadsheet

Since we are only using DB2 WLM Request Execution times, we have to assume that the other DB2 processes that are not accounted for in WLM. The following list is a partial list of entities that do not work within a database and are not tracked by service classes:

- ▶ DB2 system controllers (db2sysc)
- ▶ IPC listeners (db2ipccm)
- ▶ TCP listeners (db2tcpcm)
- ▶ FCM daemons (db2fcms, db2fcmr)
- ▶ DB2 resynchronization agents (db2resync)
- ▶ Idle agents (agents with no database association)
- ▶ Instance attachment agents

- ▶ Gateway agents
- ▶ All other instance-level EDUs

Additionally, we did not account for any three predefined default service super classes: the default user class, the default maintenance class, and the default system class. These could be added, if you determine there is a significant difference in the overall accounting to warrant including these default super classes.

Graphically, we can represent the spreadsheet as shown in Figure 9-7 to give us a picture of how each department compares to each other.

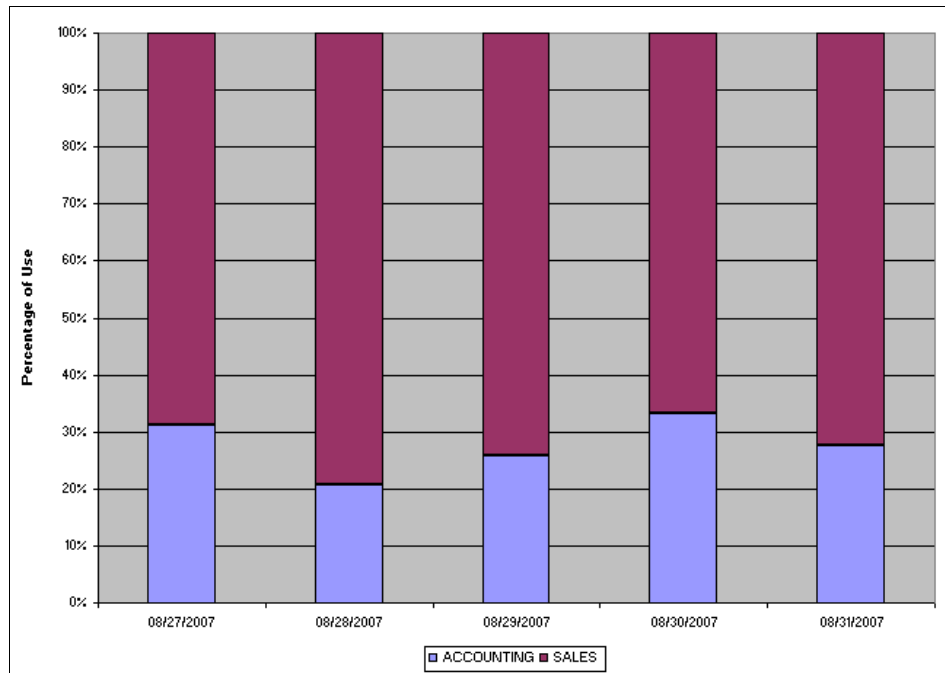


Figure 9-7 Chargeback accounting graphic representation

9.2.4 Summary

We have demonstrated that chargeback accounting can easily be done using DB2 WLM. Basic assumptions need to be made and validated to insure that processes not included in the chargeback don't adversely change the percentages. If it is determined that the non-accounted for processes are out of proportion, the default superclasses can be included in the chargeback accounting.



DB2 WLM and DWE Design Studio

IBM DB2 Warehouse Edition (DWE) provides everything you need to effectively implement a flexible and scalable data warehouse for dynamic warehousing. DWE is the premier solution for data warehousing, online transaction processing (OLTP) and mixed workloads. DB2 DWE features include enhanced warehouse management, analytic application development, OLAP, data mining, advanced compression and workload management.

In this chapter we provide a detail description of how to create DB2 WLM components using DWE Design Studio.

10.1 DB2 Warehouse Design Studio overview

DB2 DWE Design Studio in Data Warehouse version 9.5 introduces a new function to create, modify, validate and execute DB2 WLM objects. Its graphical interface let you see the hierarchy of the WLM objects and manage them.

The DB2 DWE Design Studio is based upon the open source Eclipse platform. The Eclipse platform is for building integrated development environments (IDEs). Eclipse is an open source community of companies that focuses on building development platform, runtimes, and frameworks providing a universal framework for tools integration.

The DB2 DWE Design Studio provides the core components to graphically organize organizational data (data structure) and create relationship between data elements (data mining), move and transform data within the data warehouse, analysis of data to reveal business trends and statistics, identify relationships and patterns, and create database WLM objects.

When creating new tools, developers have to only code on their expertise subject area and only need to build the features that comprise their specialty, other components like runtime environment, user interface and help systems were part of Eclipse. The additional capabilities that the tools vendors provide are delivered as a plug-in to Eclipse. The plug-in is installed into an existing Eclipse environment. Each of the capabilities that Design Studio provides are packaged together and are installed on the top of the basic Eclipse platform. The basic Eclipse architecture shown in Figure 10-1.

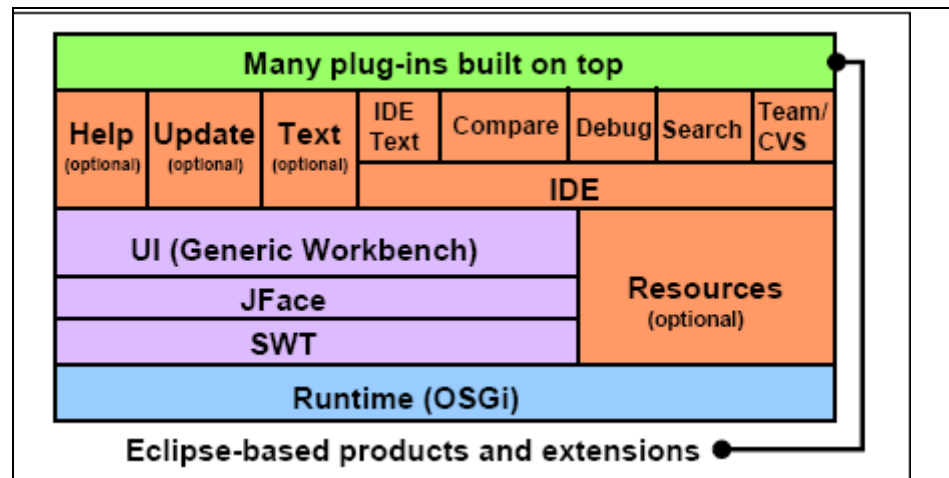


Figure 10-1 Eclipse basic platform

10.1.1 Workload management support

Prior to DB2 9.5, workload management solution was based on Query Patroller product and DB2 Governor. Query Patroller is a predictive tool, and provides a way to monitor, manage, control the work and provides reports. DB2 Governor is a reactive tool which uses system monitor to watch the work running on the system and based on the rules defined, it will take reactive steps to correct the situation.

On DB2 9.5, comprehensive workload management feature is integrated inside DB2 engine to closely interact, access and manage the workloads, so that you can see how your system is running and you can gain control over resources and performance.

DB2 DWE Design Studio 9.5 supports the ongoing development, refinement, validation, and monitoring of a workload management solution:

- ▶ Reverse engineering on existing database

You can use the Design Studio to reverse engineer an existing DB2 WLM object definitions, settings, and other informations to create a new WLM scheme. This scheme can be changed to suite your needs and validated. Reverse engineering allows you to port the current database settings to Design Studio for further editing, port to another system, or as the base for a new workload.

- ▶ Create new WLM solutions using templates

Design Studio provides standard templates which can used to achieve your WLM objectives:

- Resource sharing

On a shared system environment, sources has to be shared. Using WLM solution you can control and share the system resources based on the needs, priority, or other defined agreements. Explicitly establish limits for the consumption of system resources, prevent the misuse of resources, and track resource usage patterns. For example, you can setup CPU sharing on AIX systems and for others you can set thread priorities of the agents, and control the priority with which agents submit prefetch requests.

- Enforcing database activity limits

Limits can be enforced in the database by specifying whether the activities that exceed the boundaries are queued, stopped, or allowed to execute.

- Enforcing limits for concurrent activities

You can enforce limits on number of coordinator activities that run simultaneously. Limits can be enforced in the database, in superclass, specific type of work, or specific type of work from a specific source.

- ▶ Manipulate WLM entities

On the Design studio, superclass, workloads, work actions, thresholds can be viewed and changed using graphical interface.

- ▶ Validate of WLM entities

After creating a workload management scheme, you can validate it to ensure that you have entered values for the required properties and defined all of the property values correctly.

If the validation fails, the Design Studio displays error and warning messages output to alert you of the problems in the scheme that need to be corrected.

On successful validation, you can generate the SQL code, review it, and revise the scheme to change the code. This can be repeated till you are satisfied that the scheme is ready to be deployed to the database.

- ▶ Deployment

After a successful validation, you can deploy the workload management scheme directly from Design Studio by connecting to the database.

10.1.2 Installing DB2 DWE Design Studio

Provide how to get DB2 DWE Design Studio information and brief description of installation process or point out where to get the installation information.

Author Comment: Waiting for document. Still not clear with Install intructions from the DataWarehouse.

Note: Eclipse platform is installed as part of Design Studio installation process, separate Eclipse download and install is not required.

10.2 Getting start

To start the Design Studio, (provide how to start info).

Once the Design Studio is started, you are prompted to enter the workspace path as shown in Figure 10-2. A workspace is place where you can store all your Design Studio work and data files. It acts as a central repository for your data files. A workspace may hold multiple projects. You can have more than one workspace but only one will be active per running instance of DB2 DWE Design Studio.

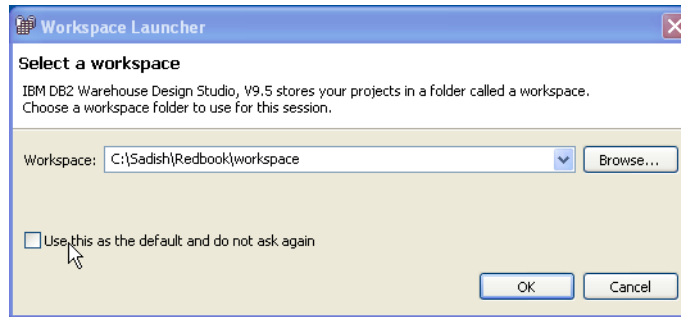


Figure 10-2 Workspace location

To switch between workspace, select **File** → **Switch Workspace**.

Note: We recommend that you select the workspace path which will be frequently backed up.

After you specify the workspace, the Design Studio displays the Welcome view, which includes links to the DB2 Warehouse documentation, tutorials, and sample files.

Design studio work are stored as projects, files, and folders. A project is a container used to organize resources pertaining to a specific subject area, for example data warehouse project. The workspace resources are displayed in a tree structure with projects containing folders and files. Projects may not contain other projects.

Figure 10-3 shows the DWE Design Studio workbench.

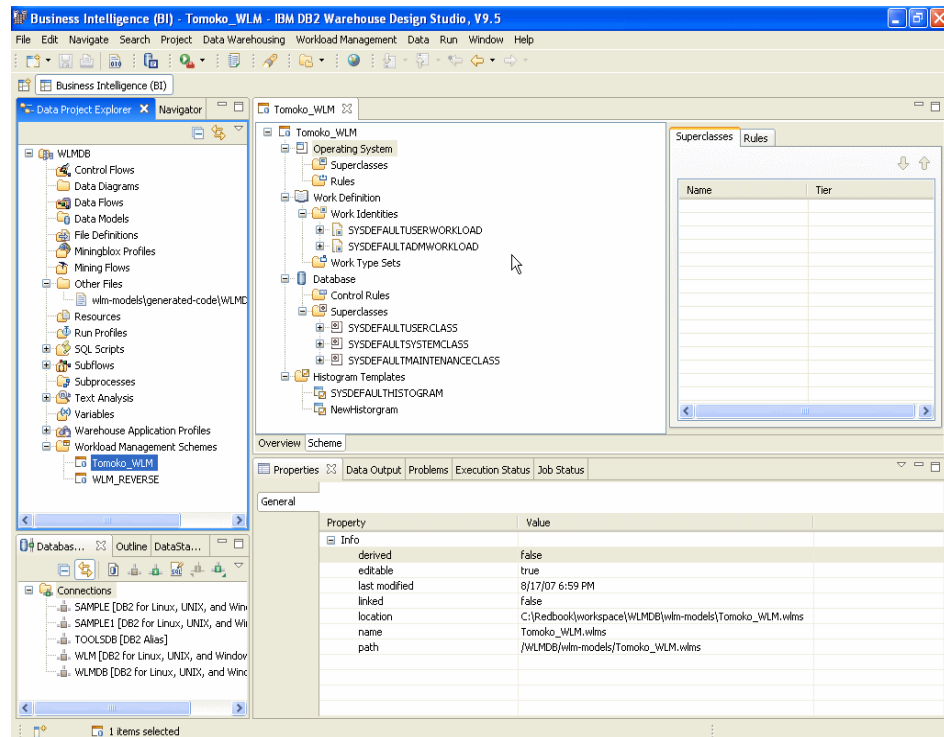


Figure 10-3 DWE Design Studio workbench

Each panel in the Design Studio contains one or more perspectives that contain views and editors. The perspectives control what the Design Studio displays certain menus and tool bars.

Perspectives

When you open the Design Studio, it takes you to the default BI perspective. This view contains various resources that can be used to accomplish a particular task or work with a particular resource. Certain menu options are enabled or disabled based on your current perspective. Additional perspectives can be opened based on needs. Perspectives provide the functionality required to accomplish a particular task with a particular resource.

To open additional perspectives click **Window** → **Open Perspective** → **Other** and select desired perspective.

To reset a perspective to its original layout, click **Window** → **Reset Perspective**

Views

A view is a visual component of the Design Studio to display properties, tree structure and access to editors. Views can be used to navigate Data Project Explorer and Database Explorer information trees, open the related editors, or display and review properties. When you modify the information in a view, the Design Studio saves it immediately. To open a closed or hidden view, click **Window** → **Show View** and select the view.

data project explorer view

This view is open by default in the upper, left area of the Design Studio. This hierarchical tree displays projects and objects that you can navigate through or create new objects. This is one of the most frequently used view where you select and modify the contents of different projects and objects. This view is not *live* and does not provide direct access to underlying databases.

Database explorer view

This view is open by default in the lower, left area of the Design Studio. It is a hierarchical tree of the live databases that you can explore, connect to, create models from, and ultimately modify. You must have a DB2 user account that includes the appropriate authority and privileges to modify a database. The DB2 databases (and aliases) are listed automatically in this view which is picked up from your local catalog. You can set up connections to other databases as needed.

Properties view

This view open by default in the lower, right area of the Design Studio. You can use the properties view to define and modify many of the objects that you create. In addition to this, you can see other views such as Data Output, Problems, Execution Status and Job Status. To make any of these views active, click on title tab, which brings the properties view to the foreground. You can use the Properties view to define and modify many of the objects that you create. To open the Properties view when it is closed or hidden, click **Window** → **Show View** → **Properties**.

Note: If you cannot find an option or control that you expected to work with, ensure that you have opened the correct view.

Editors view

The editor opens by default in the upper, right area of the Design Studio canvas. It also opens up a palette for graphic editors, on the right side of the canvas based on the object type you are working with.

An editor is a visual component of the Design Studio that you typically use to browse or modify a resource, such as an object in a project. The available options are Text Editor, System Editor (operating system), In-line Editor based on the project scheme or objects.

Note: There is no automatic save option for these editors, you must explicitly save the changes.

Projects

A project is a set of objects that you create in the Design Studio as part of the data transformation or warehouse building processes. You must create a project in Design Studio before starting creating WLM objects. Each project that you build is represented by an icon in the Data Project Explorer, where you can expand it, explore its contents, and access editors to work with it. You create different types of objects according to the type of project you are building.

You can integrate the project file workspace directory with concurrent versions system (CVS). In a coordinated development environment this will be useful to share the project with other developers.

10.2.1 Workload Management Scheme

Workload Management Scheme supports the ongoing development, refinement, validation, and monitoring of a workload management solution. It contains projects and templates and provides a sequence of screens to guide you through for building DB2 workload management objects. You also can integrate DB2 workloads with operating system workloads. Currently, the supported operating system is AIX workloads.

There are some terminology differences between the Workload Management Scheme on Design Studio and DB2 Workload Management. Table 10-1 lists the mapping of terminologies.

Table 10-1 *Design Studio Workload entity and DB2 Workload Manager objects*

Workload Management entity in Design Studio	DB2 Workload Manager object created in DB2
Superclass	Service superclass
Subclass	Service subclass
Work identity	Workload
Work type set	Work class set

Workload Management entity in Design Studio	DB2 Workload Manager object created in DB2
Work type	Work class
Control rule for the database	Threshold for the database domain
Control rule for a superclass	Threshold for the superclass domain
Control rule for a subclass	Threshold for the subclass domain
Control rule for a work identity	Threshold for the workload domain

On Design Studio, when you create a new scheme, it can be viewed in more than one way. You can see the DB2 WLM entities by

- ▶ Tree view
- ▶ Grid view

Figure 10-4 on page 243 shows a Design Studio tree view with WLM entities.

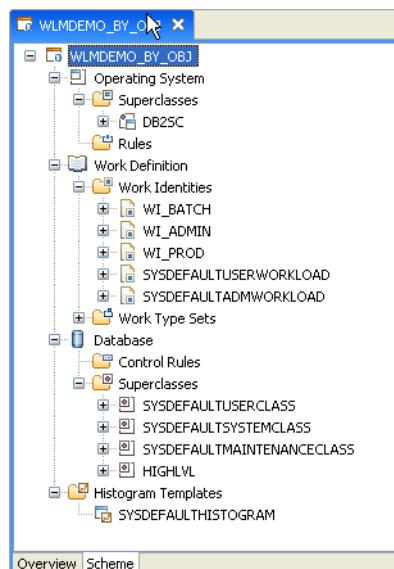


Figure 10-4 Design Studio WLM entities - tree view

Figure 10-5 shows the Design Studio WLM entities in grid view.

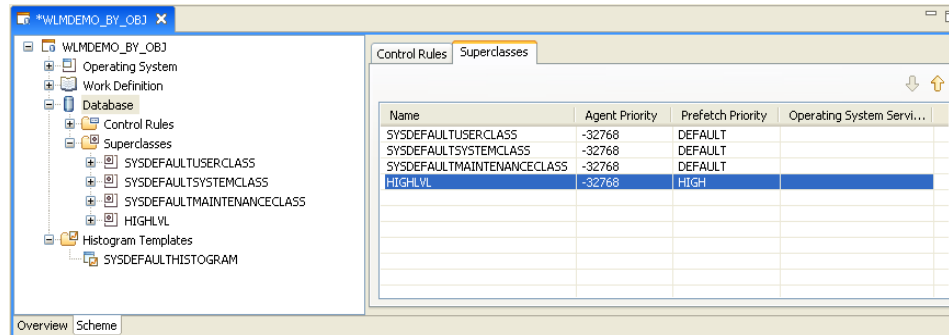


Figure 10-5 Design Studio WLM entities - grid View

When you create a new scheme, Design studio automatically creates containers for entities that make up a scheme. The containers created by Design Studio are:

► **Database service classes**

Based on the definition, all of the work for a database is executed in the database service classes.

► **Operating system service classes**

If you run the database on AIX, use operating system service classes to allocate CPU shares for database work. Operating system service classes are created with a new scheme only when reverse engineering

► **Work identities**

Work identities define the work to be controlled based on connection attributes like user or application name.

► **Work type sets and work types**

Classify the database work into sets of work types, such as DML, DDL, Read, Write, LOAD, CALL and so forth. Work types are then mapped to the database subclasses that execute the work.

Note: Work type sets and work types are never created for you as part of creating a new scheme, this has to be created separately.

► **Histogram templates**

Histogram templates are used to determine the high bin values for a histogram. You can create custom templates for the histograms that display your monitoring data, or use the default templates.

10.3 Managing database workloads using Design Studio

You can use the Design Studio to perform the DB2 Workload Management methodology to achieve your goal:

- ▶ Plan and design the workload management system
- ▶ Review and finalize your management goals
- ▶ Create and implement baseline monitoring
- ▶ Create execution environment and implement the controls
- ▶ Monitor and repeat the process till you achieve the desired goal

You can use a data warehouse project for designing and building the DB2 workload management objects. Before starting to do any work, create a project from **File** → **New** → **Data Warehouse Project**. See Figure 10-6.

In this example, we create a new project WLMDB.

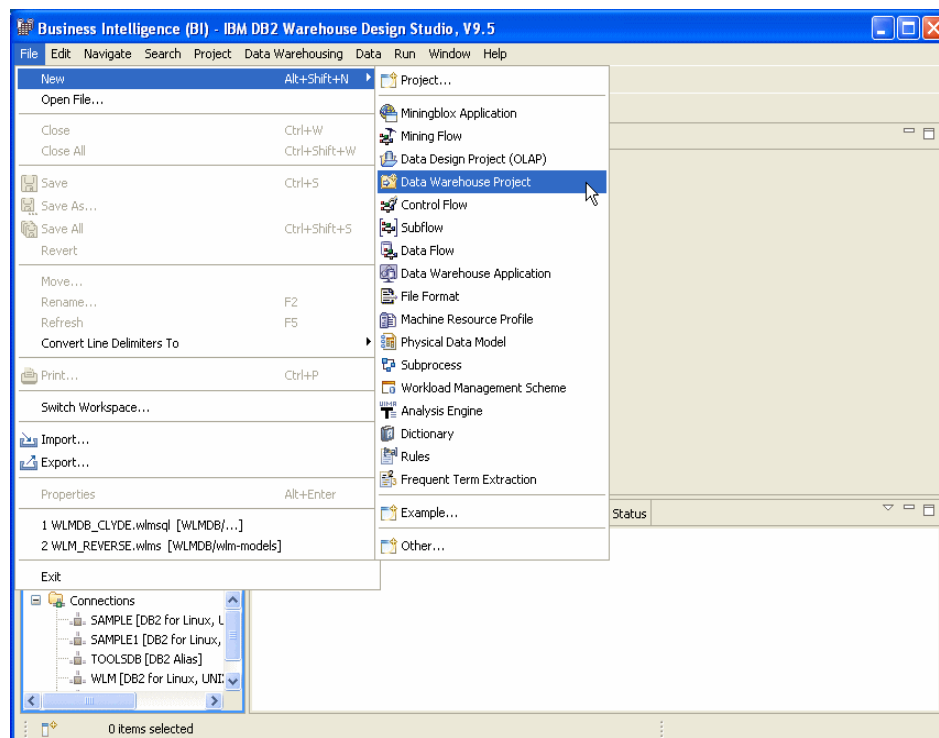


Figure 10-6 Create a new Data Warehouse project

After creating a data warehouse project, create a new workload management (WLM) scheme. The WLM scheme contains a set of entities that define a workload management solution for a database. When you create and save a WLM scheme for the first time, Design Studio create a file with `.wlms` file extension for the scheme using the scheme name you specified under the workspace directory. The Design Studio displays the complete path in Overview tab.

To create WLM Scheme, expand the project tree view (WLMDB in our example), right click **Workload management scheme** → **New** → **Workload Management Scheme**. See Figure 10-7 on page 246.

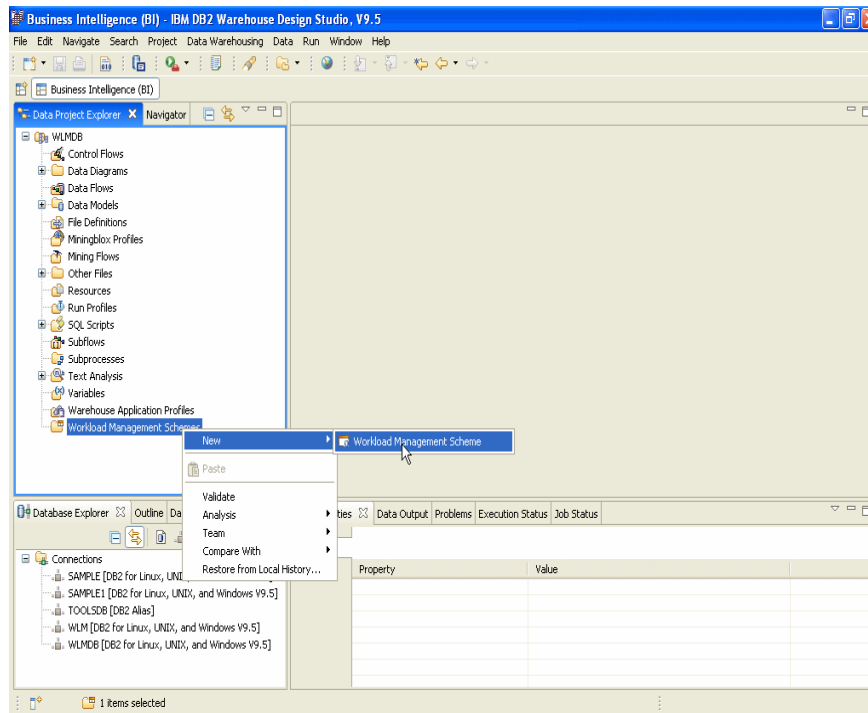


Figure 10-7 Creating workload management scheme

When you select New Workload Management Scheme, it takes you to New File for Workload Management Scheme panel (Figure 10-8 on page 247) where you can give a scheme name and select one of the three methods to create the scheme:

- ▶ Create a scheme by objective
- ▶ Create a scheme yourself
- ▶ Create a scheme by reverse engineering.

We discuss each scheme creating methods in detail in the following sections.

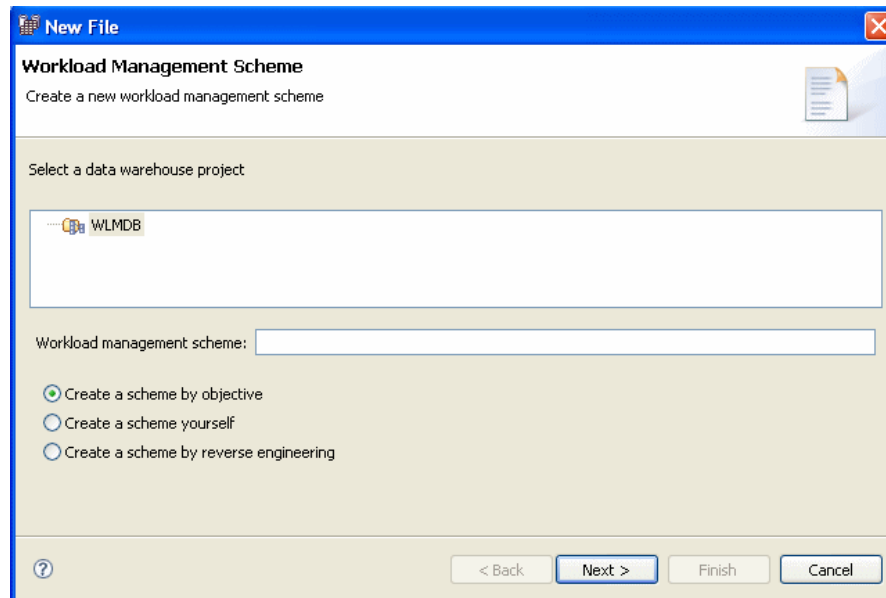


Figure 10-8 Select creating workload management scheme method

10.3.1 Create workload scheme by objective

When you create a WLM scheme by objective, Design Studio guides you through the configuration process. This method helps to simplify the task of creating a WLM scheme. By using this method, you can create a scheme that resolves three common WLM objectives:

- ▶ Controlling and sharing system resources

This is for creating a WLM scheme to manage the system resources for database activities based on your resource allocation objectives. If the database runs on the AIX operating system, you can integrate operating system service classes to manage the system resources for the execution environments. For non-AIX systems, you can set the Agent Priority.
- ▶ Creating limits for database activities

This is for creating a WLM scheme to define and enforce execution limits on the database activities. You can specify whether to stop the activities that reach the limits or allow them to continue. You can create limits based on work type, activity, source, or combination of these.
- ▶ Creating limits for database activities that run concurrently

This is for creating a WLM scheme to create and enforce concurrency limits on database activities. You can specify whether to stop activities that exceed the concurrency limits or allows them to continue.

Controlling and sharing system resource

In this section, we demonstrate how to use *Control and share system resources* to create a workload scheme. The example business problem used here is a database system which has small, medium, and long running queries coming from different business units and compete for resources. We need to define a method so that works are grouped by business units and share system resources among each other in a controlled way.

One of the solution for this business case is

- ▶ Categorize work and create appropriate work identities.
- ▶ Based on business units create superclasses.
- ▶ Define relationships and assign different types of work identities to appropriate superclasses.
- ▶ AIX WLM provides sophisticated management of CPU. If AIX WLM is available and in place, then associate DB2 superclasses with AIX superclasses. For non-AIX, use Agent priority

The above proposed solution can be achieved using Design Studio Controlling and Sharing system resource option.

To create a WLM scheme using the Design Studio guided steps, we specify WLMDEMO_BY_OBJ as the scheme name and select **Create a scheme by objective** in New File Workload Management Scheme panel. See Figure 10-9 on page 249.

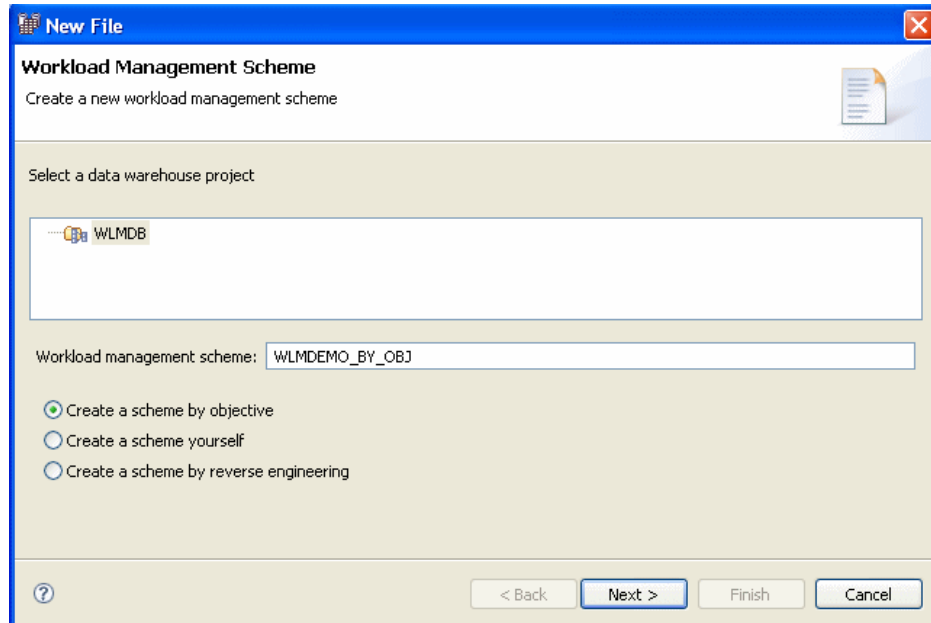


Figure 10-9 Create scheme by objective

In Create a Workload Management Scheme by Objective panel (Figure 10-10 on page 250) select **Control and share system resources** and click **Finish**.

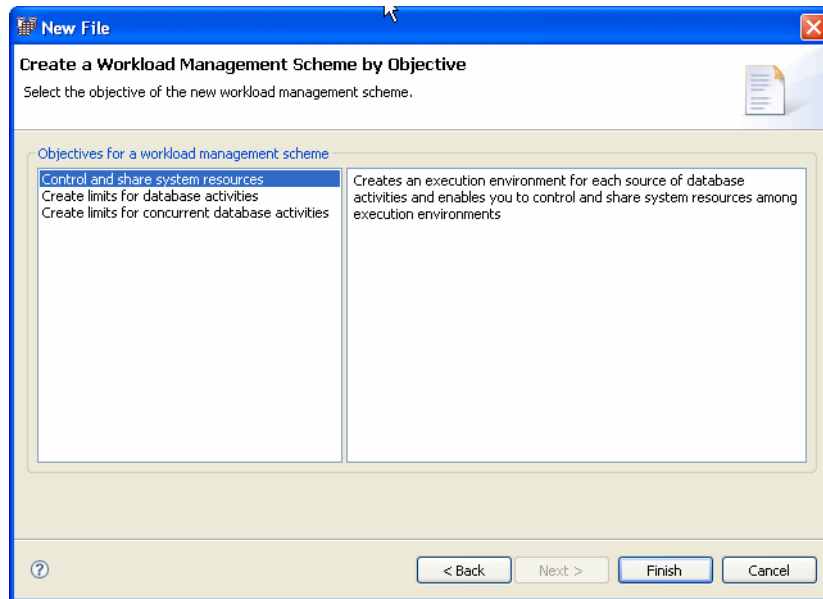


Figure 10-10 Control and share system resources

The Control and Share System Resources panel (Figure 10-11 on page 251) is presented with four tabs, General, Work Identities, Superclasses, and Create Relationships. Each tab allows you to create specific DB2 workload objects.

Note: Clicking Finish at any tab takes you back to Business Intelligence view, not to next tab. Design Studio will create only the default objects for those unvisited tabs. To continue working on the scheme using the guided configuration, select the WLM scheme → **Workload Management**, select the guided configuration you want.

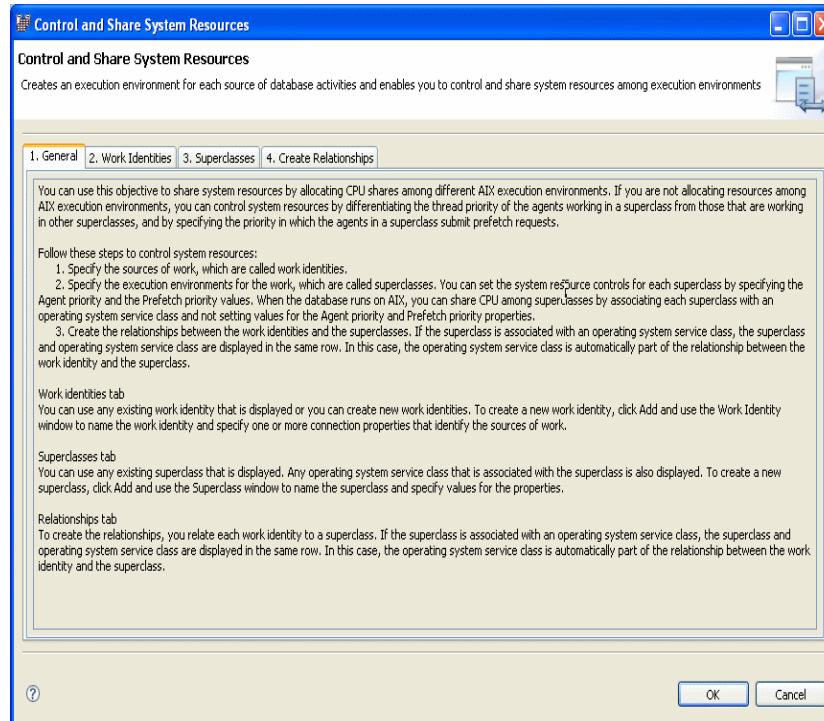


Figure 10-11 Control and share resources - Tab view

1. General: This is an informational tab. Select Work Identities tab to continue.
2. Work Identities:

You can use this tab to add a new DB2 WLM workload, delete or modify an existing workload.

Since there are no existing user-defined workloads, the default workloads, SYSDEFAULTUSERWORKLOAD and SYSDEFAULTADMWORKLOAD, are listed. To add a workload, click **ADD** and the Work Identity property view is presented. See Figure 10-12 on page 252. You can identify the connection attributes to be used in your workload.

To see the description of a field, left clicking the field name.

The capitalized fields in the Work Identity panel match the attributes in the DB2 CREATE WORKLOAD statement. The three authorization fields are for granting the workload execution authority.

We create a workload WI_PROD to manage application dss.exe. The workload can be used by everyone (public). Select **Superclasses** tab to proceed.

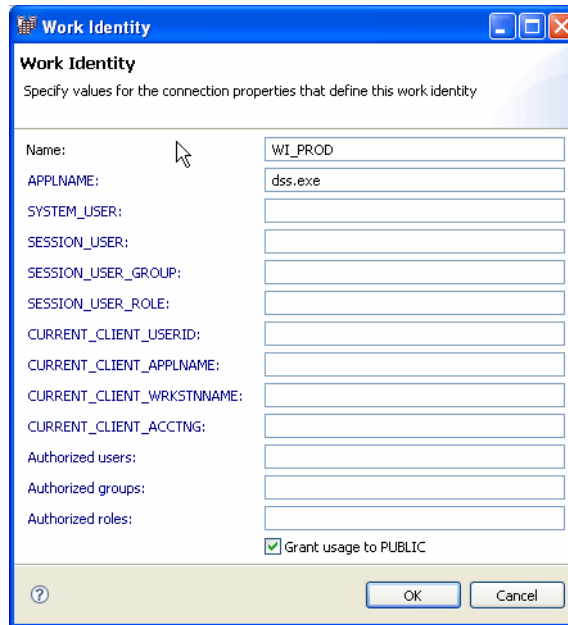


Figure 10-12 Work Identity

Note:

- ▶ If you do not specify a value for a connection property, DB2 matches for all of the values of the property, as if you had specified a wildcard.
- ▶ When you specify values for multiple connection properties, such as application name and system user, the values are interpreted as the application name AND the system user. For example, if you enter 'dss.exe' in APPLNAME and 'Bob' in SYSTEM_USER, the values are interpreted as 'dss.exe' + 'Bob' as connection property.
- ▶ If you type multiple values for the same connection property in a comma-separated list, each value in the comma-separated list is connected to the next by *or*. For example, if you enter Mary, Bob, John in the SESSION_USER field, the values are interpreted as Mary or Bob or John.

3. Superclasses:

Figure 10-13 on page 253 shows the Superclasses tab with three default superclasses have been created SYSDEFAULTUSERCLASS,

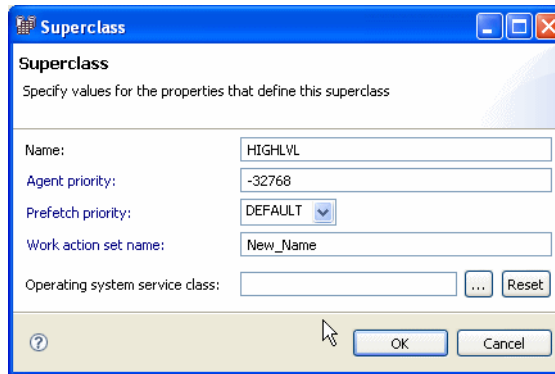



Figure 10-14 Superclass properties

If your operating system is AIX and you are using AIX WLM, you can associate the superclass with the Operating system class by selecting the browse button . This will take you to Select Operating system Service Class panel (Figure 10-15).

Note: If you associate the superclass with an operating system service class, do not set the Agent Priority property. When this parameter is set to a value, the agents are set to a priority that is equal to the normal priority *plus* agent priority when the next activity begins. For example, if the normal priority is 20 and AGENT PRIORITY is set to -10, the priority of agents in the service class is set to $20 - 10 = 10$.

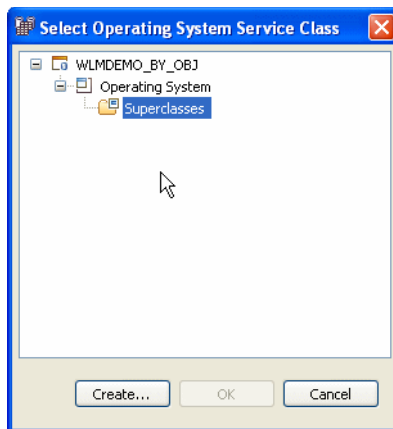


Figure 10-15 Select Operating system service class

To create a new operating system service class, expand the WLM scheme, select **Superclass**, and click **Create ...**. For this example, we create an operating system class DB2SC for DB2 resources. see Figure 10-16. Clicking **OK** takes you back to the Select Operating System Service Class to allow you to create more operating system service classes.

Figure 10-16 New Operating system service class

Once you have completed creating OS service classes, select any one of the OS service class to make OK button active and click **OK**. Design Studio takes you back to Superclass property view showing the new OS service class associated with DB2 Superclass, see Figure 10-17.

Figure 10-17 DB2 Superclass with AIX Operating system Service class

Clicking **OK** in Superclass property view takes you back to the Superclasses tab view for creating or editing another superclass. Figure 10-18 shows DB2 superclass associate with AIX superclass. Clicking **OK** in this view ends the workload creating process. To continue, click **Create Relationships** tab.

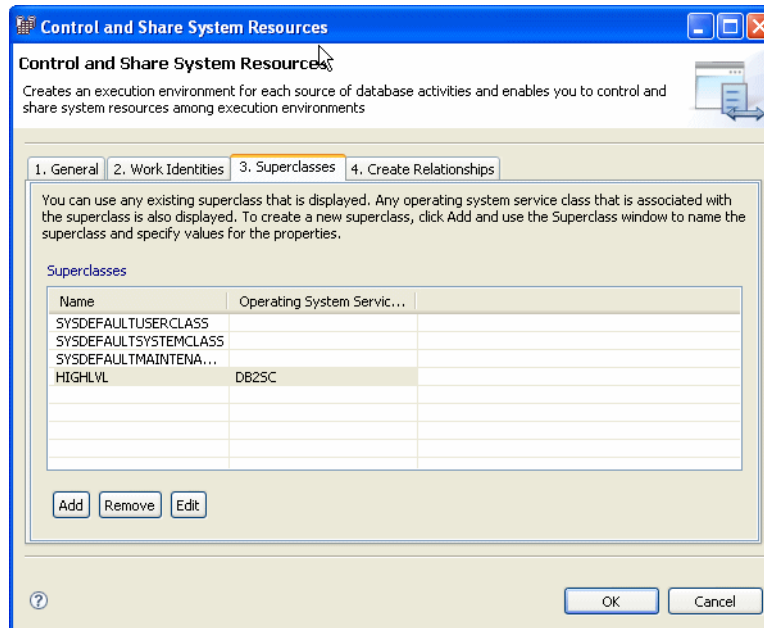


Figure 10-18 DB2 superclass and aix superclass

4. Create relationships

By default, workloads are associated with the default superclass. you can use this tab to customize the relationships between the work identities, superclasses, and operating system service classes. Figure 10-19 on page 257 shows all the defined workloads are listed under Work Identity column.

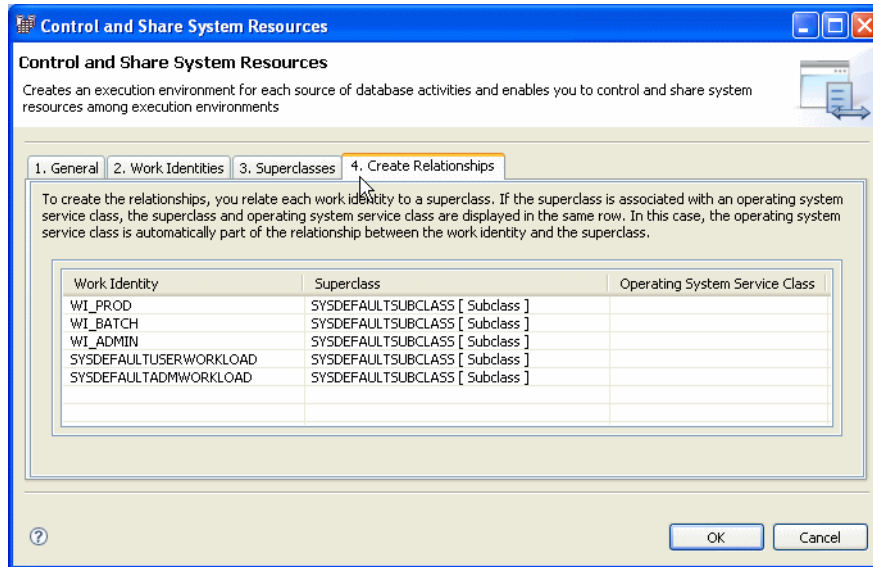


Figure 10-19 Create Relationship view

To change the superclass associated to a workload, click the superclass name, a browse button shown next to the superclass name. Click browse button to get a list of superclass you can choose from. See Figure 10-20 on page 258.

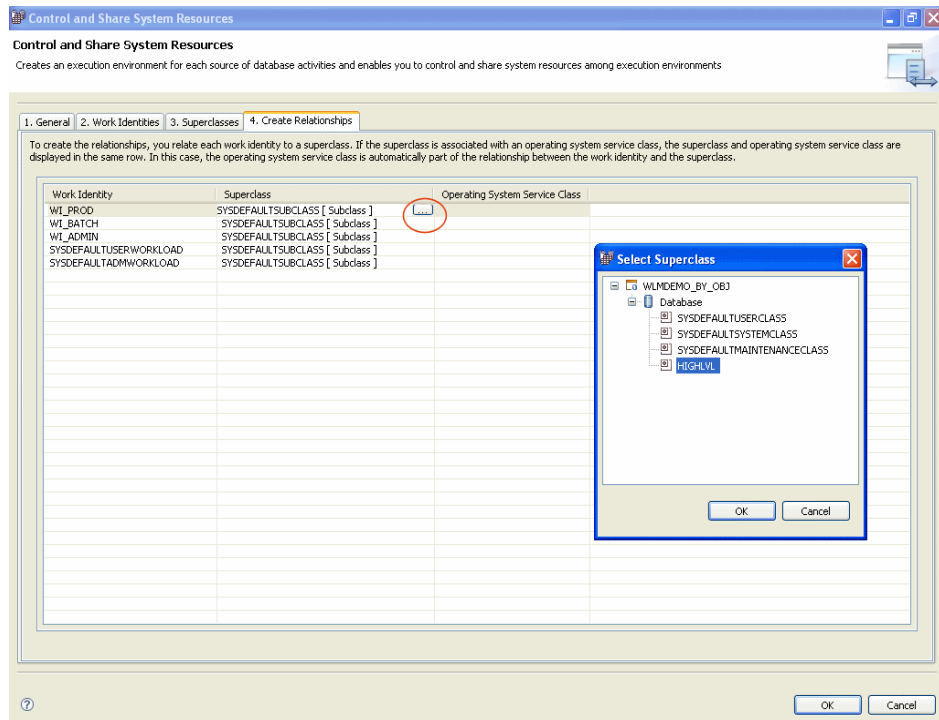


Figure 10-20 Create Relationships - Associate with superclass

If you have a DB2 superclass which is associated with an operating system service class, associating a work identity to the DB2 superclass will automatically map the work identity to the operating system service class. In our example, the DB superclass HIGHLVL was associated with the operating system service class DB2SC. Once we associate the WI_PROD to HIGHLVL, the operating system service class DB2SC is automatically associated. See Figure 10-21 on page 259.

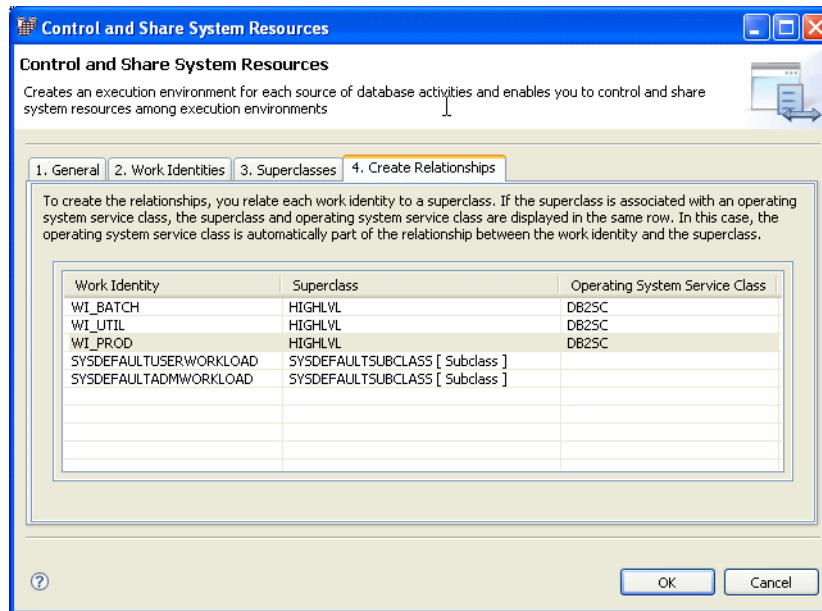



Figure 10-21 Control and share system resources - Create relationship view

When you click **OK**, Design Studio creates a WLM scheme and take you back to the default Business Intelligence (BI) perspective.

Note: On this guided Control and Share System Resources, Design Studio will not guide you through the creation of Histograms. When the project is created, a default SYSDEFAULTHISTOGRAM is created for you.

In the BI perspective, Design studio focus you on the Editors panel and give the overview of the current project created. The project created for this example is WLMDEMO_BY_OBJ as shown in Figure 10-22 on page 260.

Note: Design Studio does not save the scheme automatically. To save your scheme, use **File** → **Save** or the  icon.

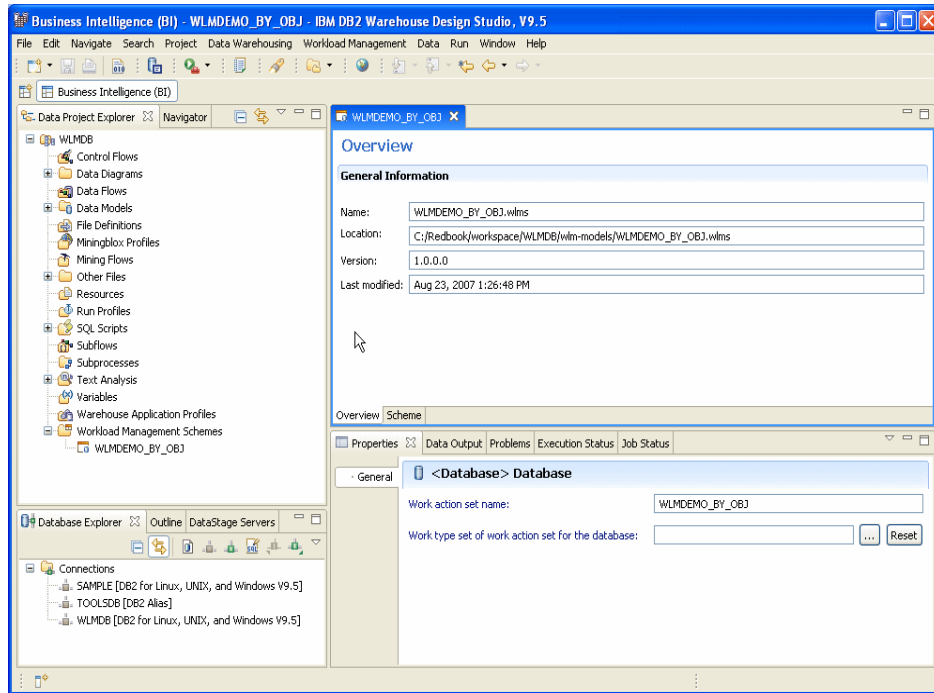


Figure 10-22 WLM Scheme created using create scheme by objective

If you select the Scheme tab in the BI perspective, you can see the tree structure of the WLM scheme you created. See Figure 10-23 on page 261. From the tree view, you can further modify the work scheme you created.

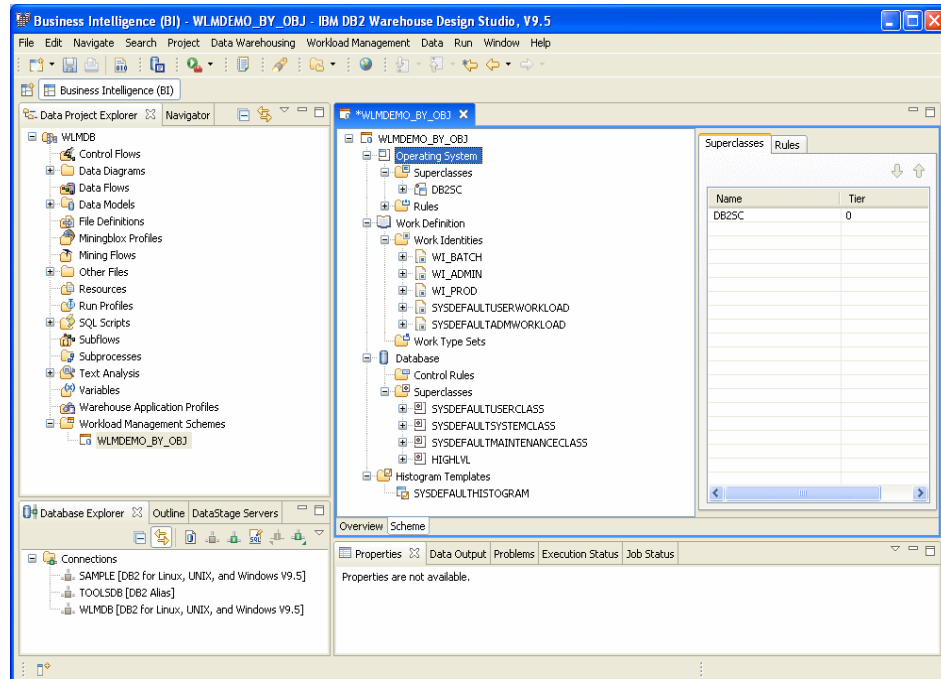


Figure 10-23 WLMDEMO_BY_OBJ Scheme view

Evaluation order of work Identities

There is an evaluation order for DB2 workloads. When a database request comes, DB2 searches the workload list in sequence to find the first one that matches the required connection properties. The search order is specified when the workload is created. In DB2, the default workload position is *last*. When you create work identities using Design Studio, it generates the code with POSITION AT *position* based on the order the work identities are created. We recommend that you verify the workload sequence before creating the workloads.

To adjust the evaluation order, open the Work Identities tab by right-clicking **Work Definitions** in the Scheme view. Select the work identity and use the up and down arrows to reposition it in the list.

Note: In Design Studio, if you do not specify an evaluation order, the order of the Work Identities tab implies the evaluation order.

Validating

Once the workload scheme is created, you can use the Design Studio Validate option to validate the workload, service classes, and the relationship you just

created. Design Studio validate all the resources on the selected project using validation settings.

To validate the WLM scheme, select **Workload Management** → **Validate**. When the validation of a WLM scheme succeeds, the Design Studio displays a confirmation message as shown in Figure 10-24 on page 262.

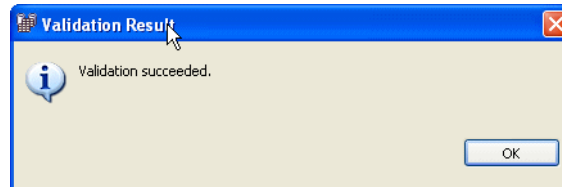


Figure 10-24 Validation successful

Note: To change the validation settings, use **Window** → **Preferences** → **Validation**

Generating code

After successful validation you can generate code using **Workload Management** → **Generate Code**. For each code file generated, Design Studio presents it as a tab in the BI perspective editor view. In our example, two files, WLMDEMO_BY_OBJ.wlmsql and WLMDEMO_BY_OBJ.osclasses, are generated. When you save, by default, the code files will be stored in the <workspace>/<schemename>/wlm-models/generated-code folder.

Figure 10-25 on page 263 shows the generated WLMDEMO_BY_OBJ.wlmsql creating DB2 Workloads.

Though the editor view allows you to add, modify, or remove the DB2 WLM statements, we donot recommend you modify the code directly since the modification will not be captured in the Design Studio. To capture them in the Design Studio, you'll need to do reverse engineer after the workload were created in DB2.

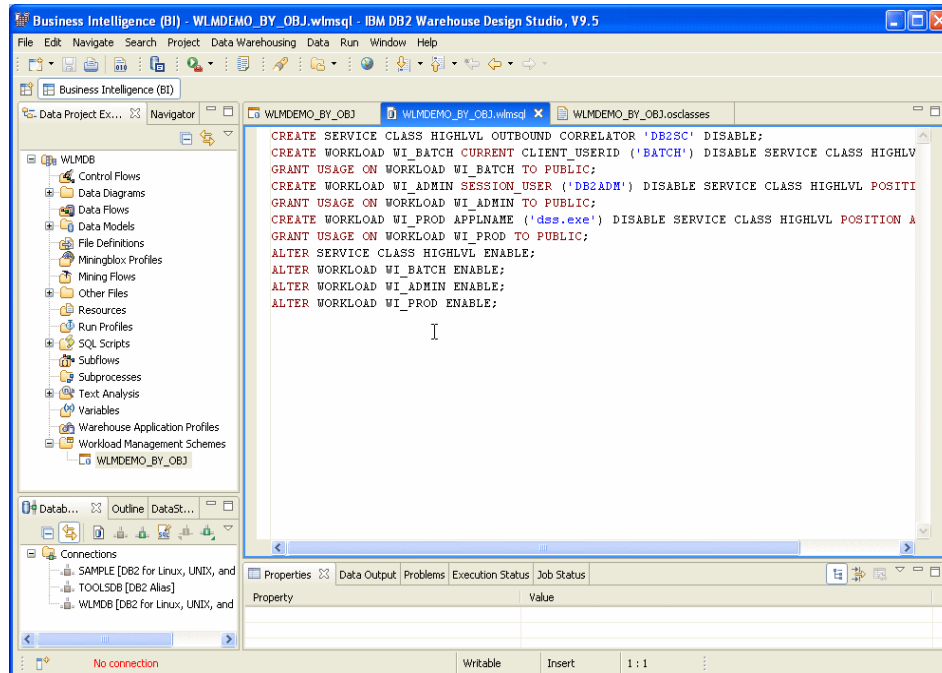


Figure 10-25 Generated Code for DB2

Figure 10-26 on page 264 shows the AIX WLM class generated under WLMDEMO_BY_OBJ.osclasses.

Note: Any OS WLM entities you set up using Design Studio will *not* be automatically created on the target AIX machine. Users have to manually copy generated OS code to AIX machine and run with root authority. Only DB2 WLM entities will be automatically created when you perform Execute or Delta Execute from Design Studio.

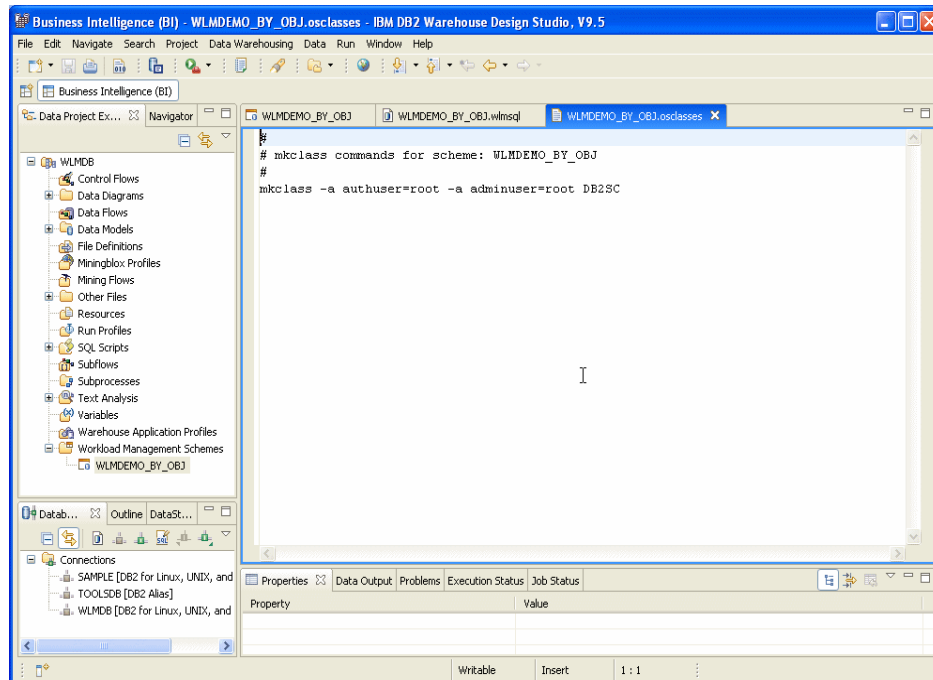


Figure 10-26 Code generation for OS WLM

Create limits for database activities

In this section, we demonstrate how to use *Create limits for database activities* to create the WLM scheme that can manage poor database activities which degrade overall performance. For example, a database system that has reports run by Sales department which usually run in five to ten minutes. On one occasion, a weekly sales report ran for six hours. Your reporting application has a built-in timeout limit. The application waits up to ten minutes for a query to return data, and then displays an error message. You want to stop the query from executing when the application displays this error message. You also want to collect detailed data about the problems that cause the application error.

One of the solution is :

- ▶ Categorize work and create appropriate work identities
- ▶ Categorize the bad queries by defining control rules on work identity.
- ▶ Specify actions for the Control Rules. The actions can be
 - STOP EXECUTION
 - Collect ACTIVITY DATA and CONTINUE

After creating project, create WLM scheme by **Workload Management Schemes** → **New** → **Workload Management Scheme**. In the New File - Create

a Workload Management Scheme by Objective panel, select **Create a scheme by objective** and click **Finish**. In Create a Workload Management Scheme by Objective panel, select **Creating limits for database activities** (See Figure 10-27) and click **Finish**.

Note: Design studio provide you an alternate way to go to Create limits for database activities screen. First, select the Database management scheme you like to work on and then do **Workload Management** → **Create Limits for Activities**

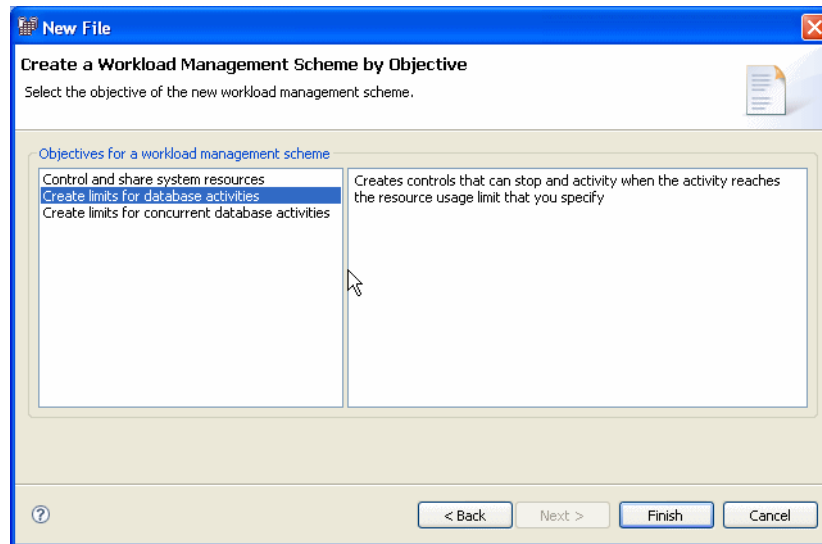


Figure 10-27 Create limits for database activities selection screen

The Create limits for database activities is presented with five tabs. See Figure 10-28 on page 266.

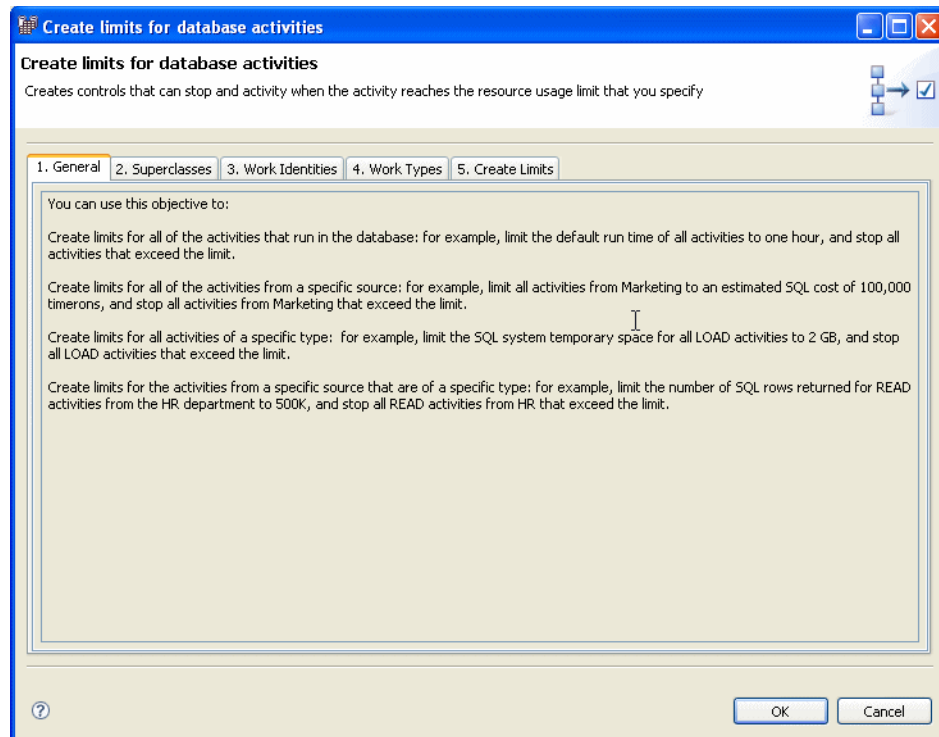


Figure 10-28 Create limits for database activities - General tab

1. General:

This is an informational tab. Select **Superclasses** tab to continue.

2. Superclasses:

Figure 10-29 shows the Superclasses tab with three default superclasses SYSDEFAULTUSERCLASS, SYSDEFAULTSYSTEMCLASS, and SYSDEFAULTMAINTENANCECLASS. Use this tab to add new superclasses, delete or modify existing superclasses.

To add a superclass, click **ADD** and the superclass property view is presented (refer to Figure 10-14 on page 254). Here you enter the name of the superclass to be created, the agent and prefetch priority, and work action set name if already know. These fields matches the AGENT PRIORITY, PREFETCH PRIORITY in DB2 CREATE SERVICE CLASS statement.

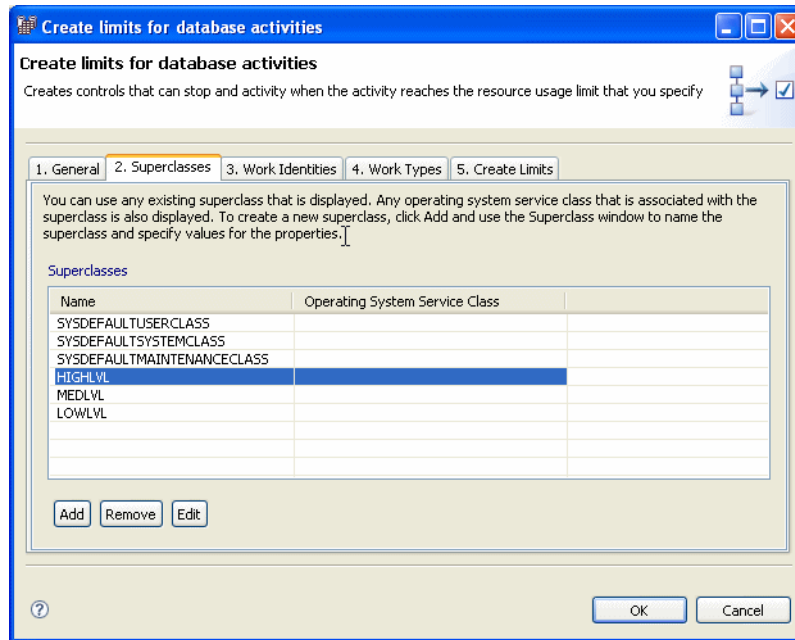


Figure 10-29 Creating superclass in Create limits for database activities

3. Work Identities:

You can use this tab to add a new DB2 WLM work identity, delete or modify an existing work identity. If there are no previous user-defined work identities created, only the default workloads, SYSDEFAULTUSERWORKLOAD and SYSDEFAULTADMWORKLOAD, are listed. To add a work identity, click **ADD** and the Work Identity property view is presented. See Figure 10-12 on page 252. You can identify the connection attributes to be used in your workload. To see the description of a field, left clicking the field name.

The capitalized fields in the Work Identity panel match the attributes in the DB2 CREATE WORKLOAD statement. The three authorization fields are for granting the workload execution authority.

We create a workload WI_PROD to manage application dss.exe. See Figure 10-30. The workload can be used by everyone (public). Select **Work Types** tab to proceed.

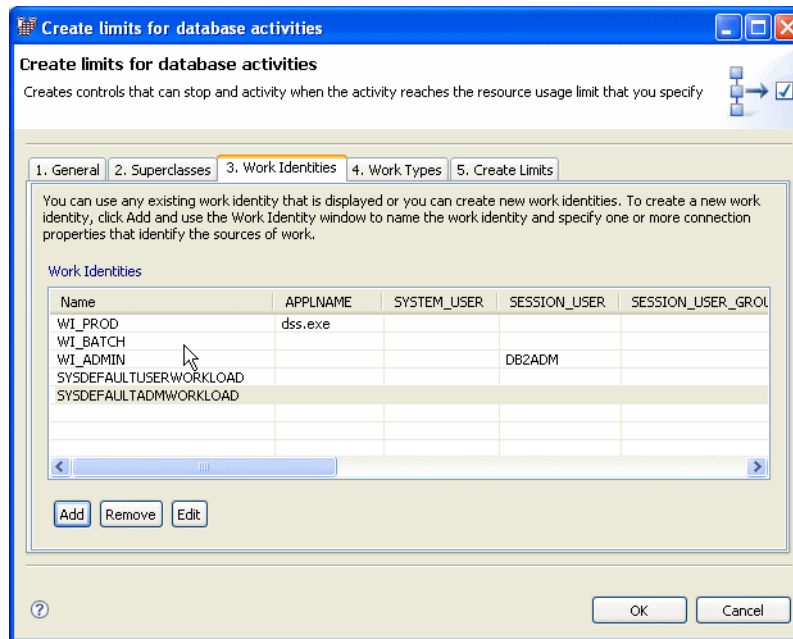


Figure 10-30 Creating work identities in create limits for database activities

4. Work Types:

You can create a work type to categorize database activities by activity characteristics type such as Read, Write, DML, DDL, Call, Load and All. Then you can manage each work type as a unit. By creating a mapping rule for a database superclass, you can map a work type to the subclass where the database activities execute.

Figure 10-31 shows the Create Work Type panel. You create a work type set to contain and manage a group of work types. If no suitable work type set is available, you can create one using **Create New Work Type Set ...**

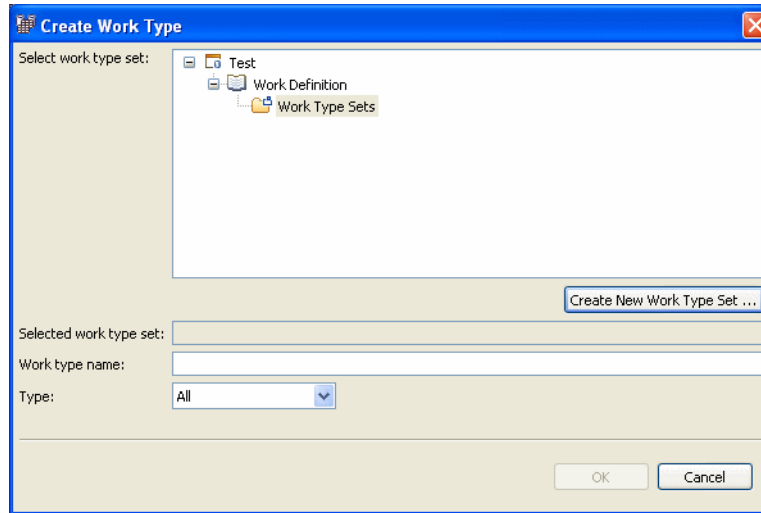


Figure 10-31 Create work type main screen

Figure 10-32 shows the **Create New Work Type Set** property view. In our example, we create a Work type set WTS_ALL.

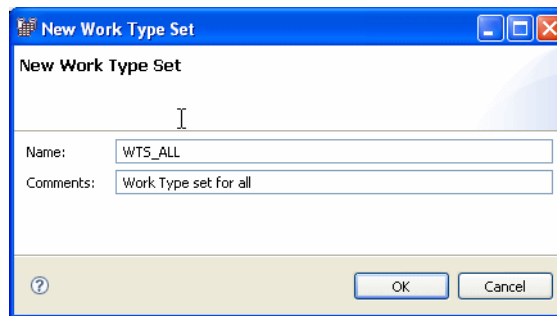


Figure 10-32 Create new work type set

To define and associate a work type with a work type set, select the work type set, enter the work type name (WT_ALL in our example), select a work type, and click **OK**. See Figure 10-33 on page 270.

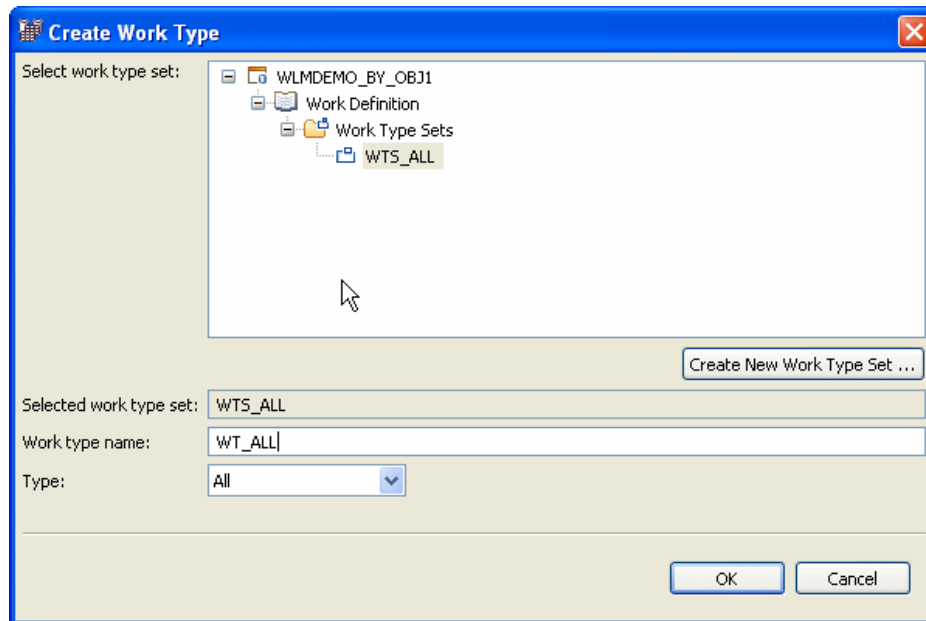


Figure 10-33 Work type set and work type association

You need to define the measurement properties for the work type. Work type optionally can include estimates. In our example, we want to apply WT_ALL regardless of the measurement, we choose **NONE** as shown in Figure 10-34.

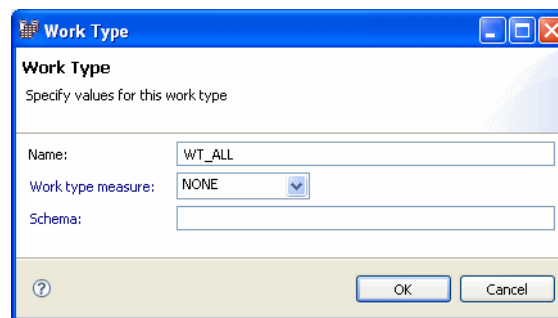


Figure 10-34 Work Type measurement properties

The capitalized field options in the Work Type Set panel and Work Type panel match the attributes in the DB2 CREATE WORK CLASS SET statement.

Figure 10-35 on page 271 shows one work type set is created. You can create more type set using the same process. After completion, select **Create Limits** tab to continue.

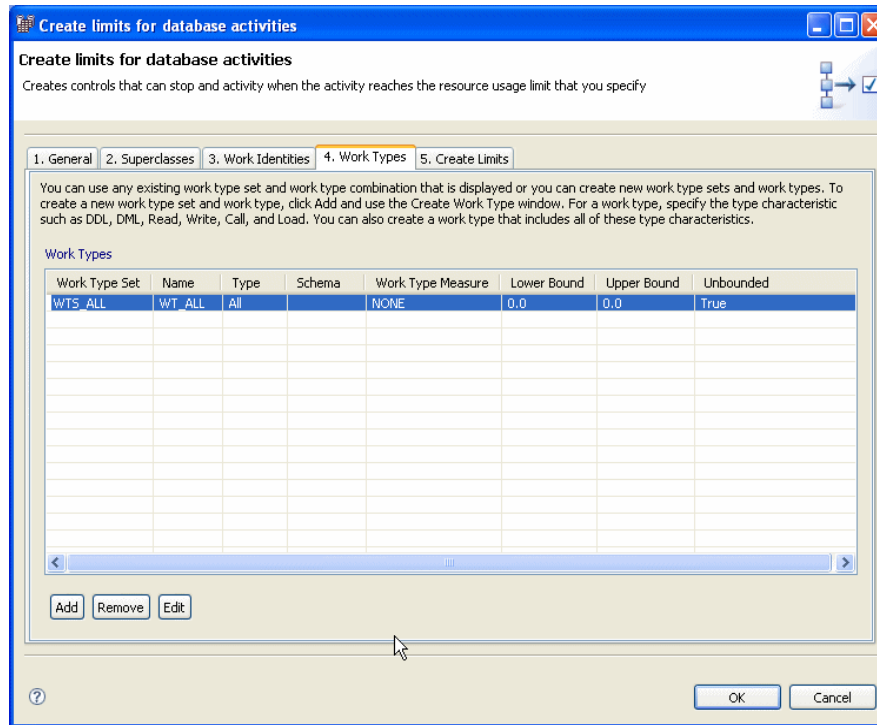


Figure 10-35 Work Type Tab after creating work type set

Note: Work types in a work type set are ordered objects when work types are evaluated. To specify the evaluation order of a work type in the work type set, open the Work Types tab by selecting the work type set in the Scheme view. Then select the work type and use the up and down arrows to reposition the work type in the list. If you do not specify an evaluation order, the order of the Work Types tab implies the evaluation order.

5. Create limit

In the Create Limits tab, click **Add**, and select the type of limit that you want to create. See Figure 10-36.

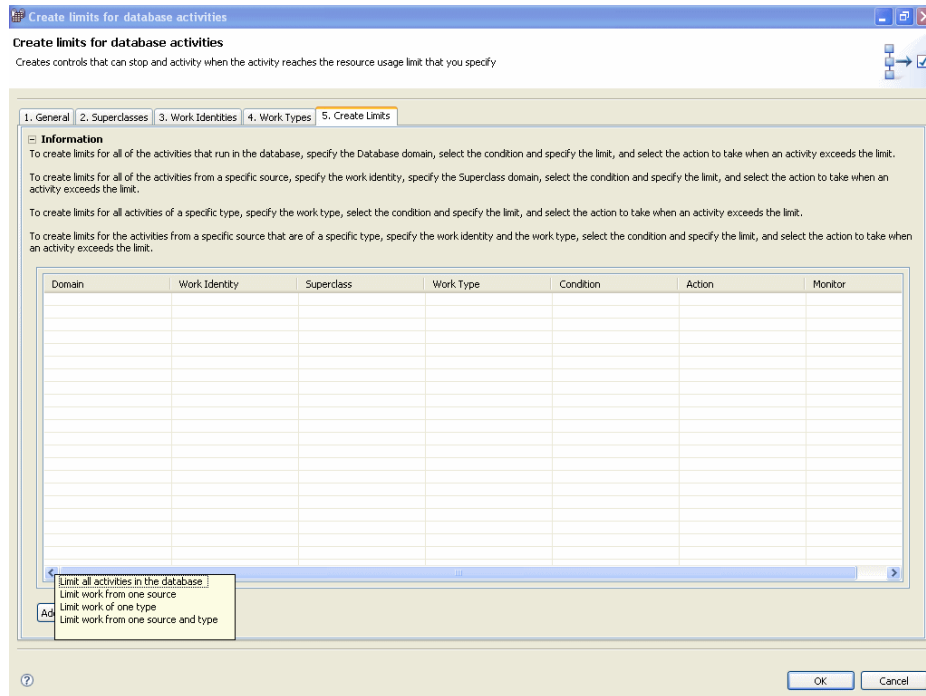


Figure 10-36 Create limits for activities

There are four options:

- Limit all activities in the database

You can use this option when limit has to be applied for database domain, and a specific action has to be performed if the limit is exceeded.

- Limit work from one source

You can use this option if the work identity is associated with a database superclass.

- Limit work of one type

You can use this option when limit has to be associated with only one work type.

- Limit work of one source and type

You can use this option, It allows you to limit work from a specific workload, of specific types (e.g. READ, WRITE, etc.).

For each option selected, Design Studio add one entry with the required fields “opened” (not greyed out). In our example, we select **Limit work of one type**

to control bad queries on database. Figure 10-37 shows for domain Work Type, the required fields are Work Type and Condition.

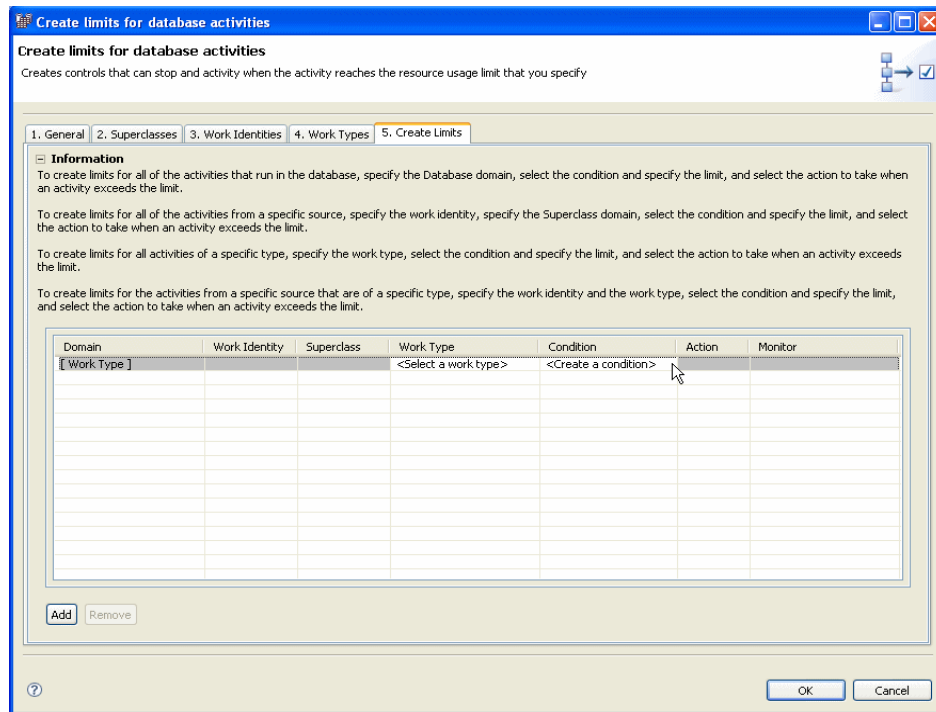


Figure 10-37 Create limits using 'Limit work of one type'

To specify the work type that you want to limit, click **<select a work type>** and a browse button shows in the field. Click the browse button to select the list of work type sets you can choose from. See Figure 10-38.

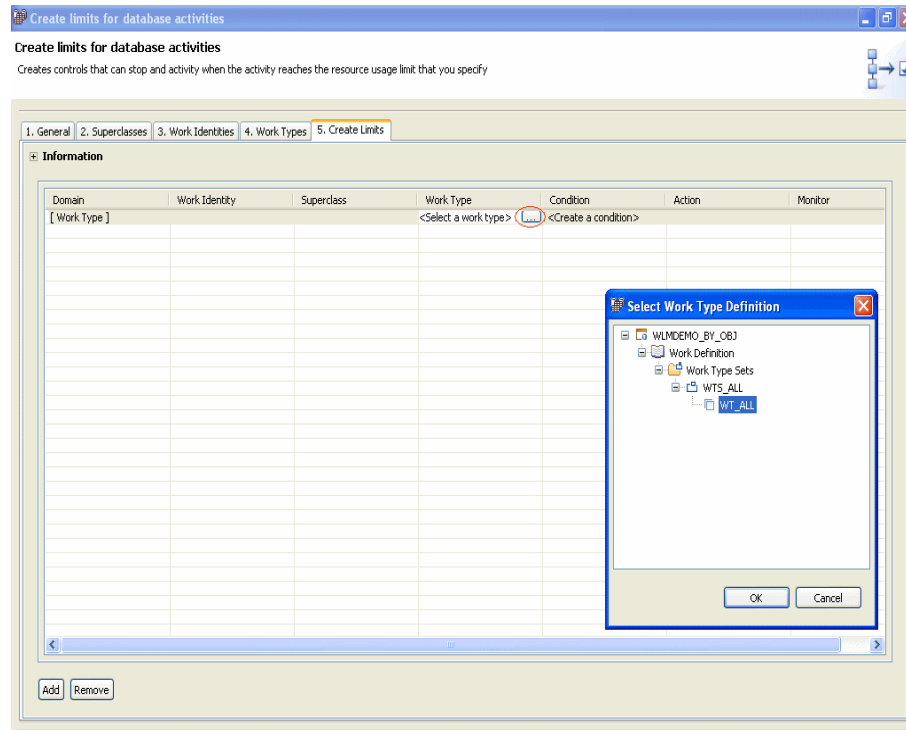


Figure 10-38 Limit work of one type - specify work type

Click **<Create a condition>** to specify the Condition for a work type that you want to limit. A the browse shows on the field. See Figure 10-39.

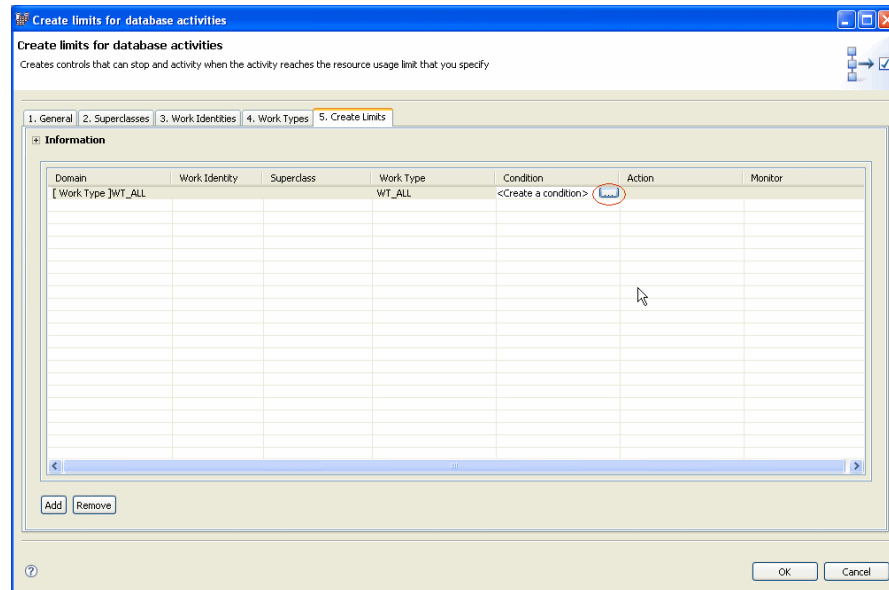


Figure 10-39 Create Limits associated with new condition

Click browse button to create a Condition to define the limitation condition on the work type domain and the action to take when the limit is exceeded. See Figure 10-40 on page 275.

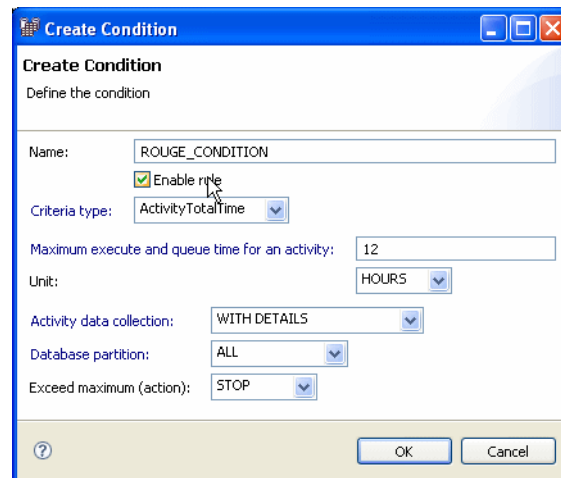


Figure 10-40 Limit work of one type - specify condition

Figure 10-41 shows that one limit is created.

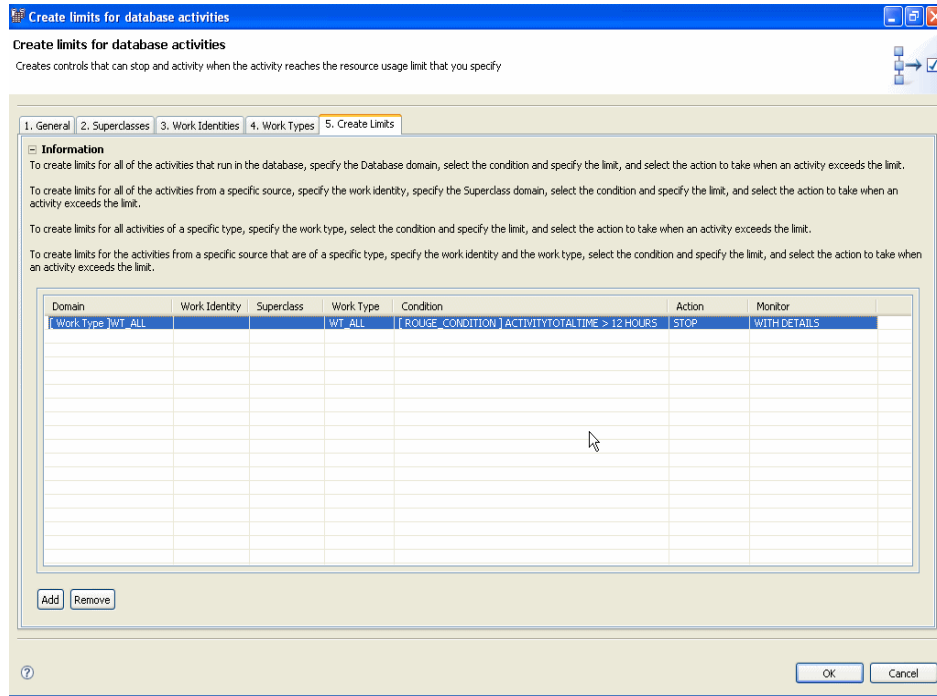


Figure 10-41 One limit created

You can continue adding limits. When you complete, click **OK** and the Database panel shown for you to associate the limits to a work action set. Click the browse button to specify the work action set. See Figure 10-42 on page 276.

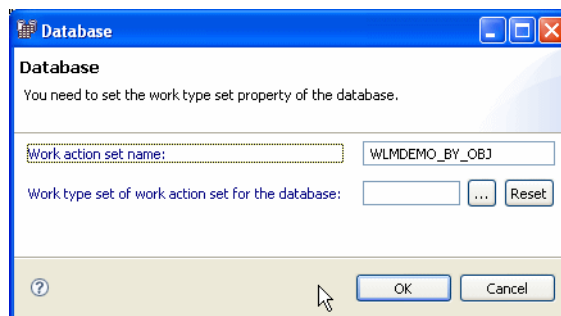


Figure 10-42 Work type set property of the database

After associating the work action, Click **OK**. Design Studio creates a WLM Scheme and take you back to the default Business Intelligence (BI) perspective.

In the BI perspective, Design Studio focuses you on the Editors panel and give the overview of the current project created. Figure 10-43 shows the project we created, WLMDEMO_BY_OBJ, and its control rules.

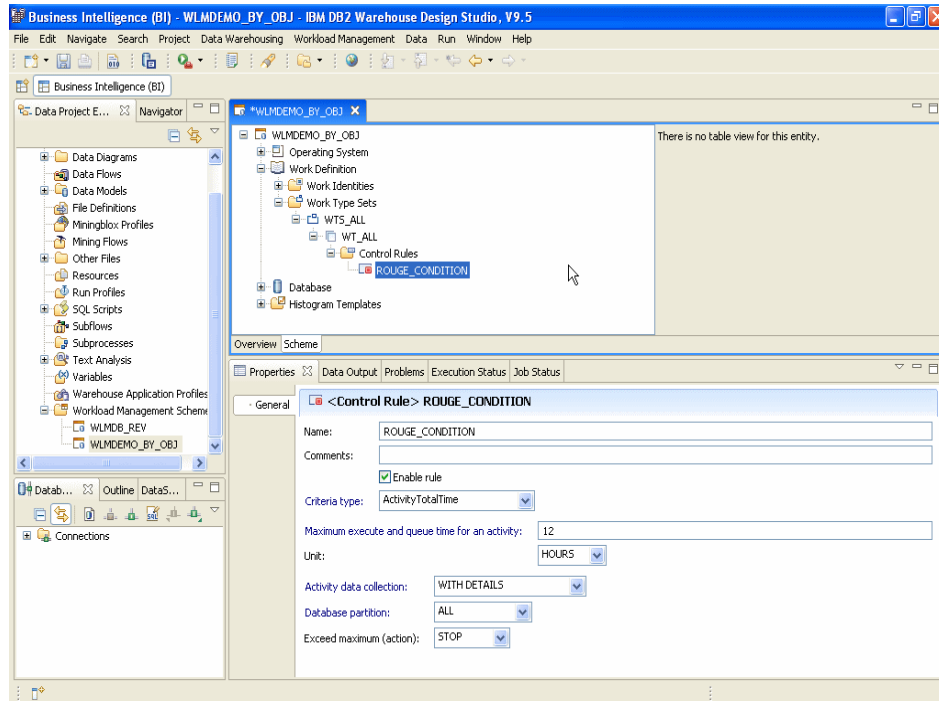
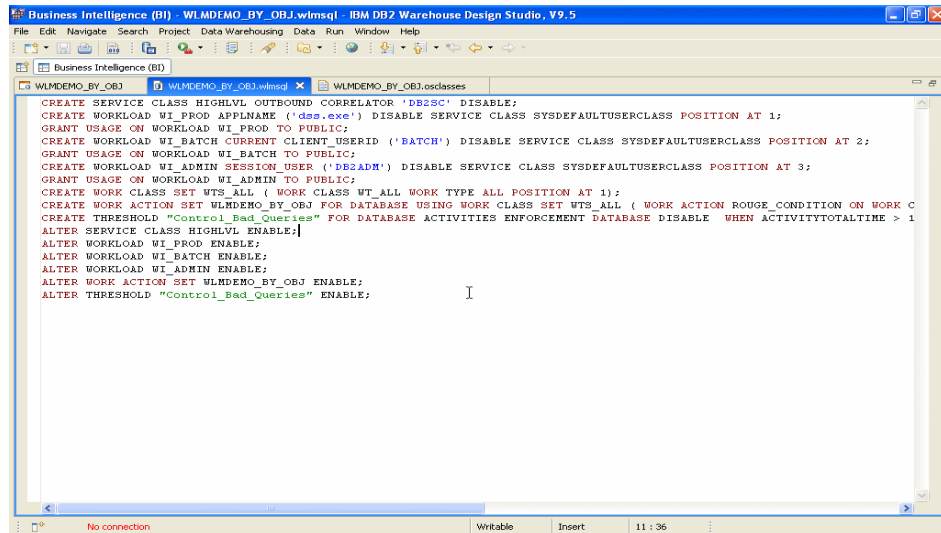


Figure 10-43 Control Rule - WLM Scheme view

Validating WLM scheme and generating code

Once the workload scheme is created, you can use the Design Studio Validate option to validate the workload, service classes, and the relationship you just created. After successful validation you can generate code using Generate Code feature. For the details, refer to “Validating” on page 261 and “Generating code” on page 262.

Figure 10-44 on page 278 shows the generated WLMDEMO_BY_OBJ.wlmsql for creating limits for database activities.



```

Business Intelligence (BI) - WLMDEMO_BY_OBJ.wlmsql - IBM DB2 Warehouse Design Studio, V9.5
File Edit Navigate Search Project Data Warehousing Data Run Window Help
Business Intelligence (BI)
WLMDEMO_BY_OBJ WLMDEMO_BY_OBJ.wlmsql WLMDEMO_BY_OBJ.osclasses
CREATE SERVICE CLASS HIGHLVL OUTBOUND CORRELATOR 'DB2SC' DISABLE;
CREATE WORKLOAD WI_PROD APPLNAME ('dss.exe') DISABLE SERVICE CLASS SYSDEFAULTUSERCLASS POSITION AT 1;
GRANT USAGE ON WORKLOAD WI_PROD TO PUBLIC;
CREATE WORKLOAD WI_BATCH CURRENT CLIENT USERID ('BATCH') DISABLE SERVICE CLASS SYSDEFAULTUSERCLASS POSITION AT 2;
GRANT USAGE ON WORKLOAD WI_BATCH TO PUBLIC;
CREATE WORKLOAD WI_ADMIN SESSION USER ('DEADM') DISABLE SERVICE CLASS SYSDEFAULTUSERCLASS POSITION AT 3;
GRANT USAGE ON WORKLOAD WI_ADMIN TO PUBLIC;
CREATE WORK CLASS SET WTS_ALL ( WORK CLASS WT_ALL WORK TYPE ALL POSITION AT 1);
CREATE WORK ACTION SET WLMDEMO_BY_OBJ FOR DATABASE USING WORK CLASS SET WTS_ALL ( WORK ACTION ROUGE_CONDITION ON WORK C
CREATE THRESHOLD "Control_Bad_Queries" FOR DATABASE ACTIVITIES ENFORCEMENT DATABASE DISABLE WHEN ACTIVITYTOTALTIME > 1
ALTER SERVICE CLASS HIGHLVL ENABLE;
ALTER WORKLOAD WI_PROD ENABLE;
ALTER WORKLOAD WI_BATCH ENABLE;
ALTER WORKLOAD WI_ADMIN ENABLE;
ALTER WORK ACTION SET WLMDEMO_BY_OBJ ENABLE;
ALTER THRESHOLD "Control_Bad_Queries" ENABLE;

```

Figure 10-44 Create limits for database activities - generated code

Design Studio guided configuration Create limits on database activities supports the following solution templates, which maps to the DB2 CREATE THRESHOLD statement.

- ▶ Create control rule on database
- ▶ Create control rule on superclass
- ▶ Create control rule for a work type (WHEN)
- ▶ Work type on a superclass that maps to a subclass with a control rule

Design Studio provide the facility to add control rules using the templates.

Adding control rules

You can add, delete, or edit control rules for an existing work identity from Scheme tab in the BI perspective. Control rules can be add or modify for work type set under Work Definition or for database. In this section, we demonstrate how to add a new database control rules.

To create a New Control Rule, expand WLM scheme and Database, right click on **Control rules**. See Figure 10-45.

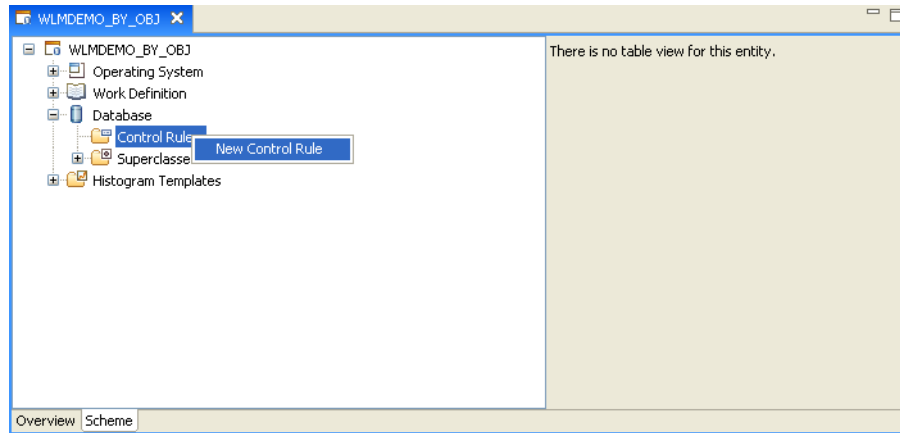


Figure 10-45 Control Rule Creation using tree view

You can complete the fields in the Properties view for the new control rule. See Figure 10-46.

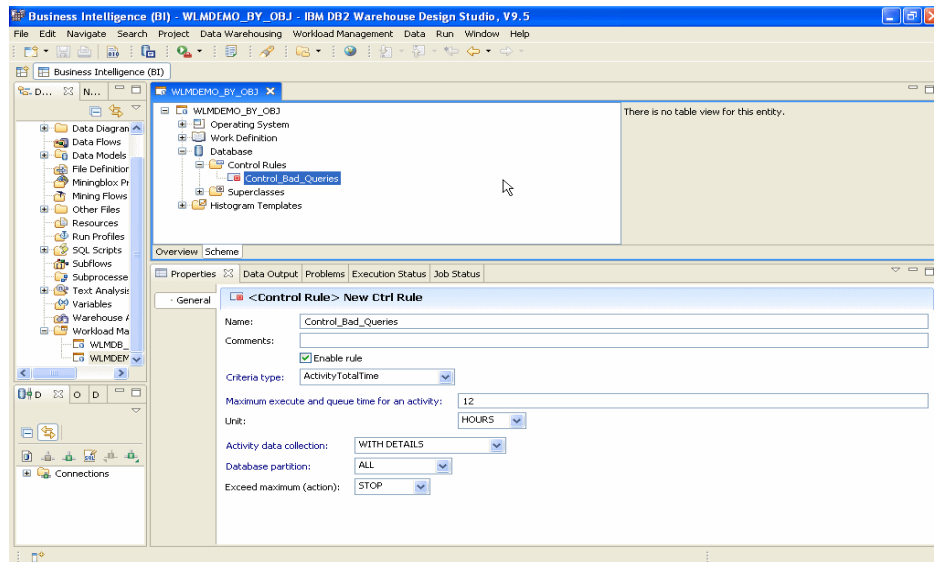


Figure 10-46 Control Rule for Database

When you create the control rule for database using Design Studio, it generates the code which is equivalent to DB2 CREATE THRESHOLD ...FOR DATABASE ACTIVITIES ...

Control rules can also be used for other tasks such as

- ▶ Force off idle connections after a specified time period.
- ▶ Apply control rule based on activity type
 - Stop activity that consume excessive temporary space
 - Allow one activity to take higher share of resources than other activities. For example, allow LOAD to use more temporary space than others.
 - Collect activity data only on higher cost queries.

Create limits for concurrent database activities

You can use the *Create limits for concurrent activities* objective to create and enforce concurrency limits on database activities. You also specify whether to stop activities that cause the concurrency limits to be exceeded or allow them to continue to execute.

An example is that the database administrator noticed that there were many simultaneous activities happening on a specific work identity and would like to do the following:

- ▶ Create a control rule to limit the maximum number of coordinator and nested activities that can execute concurrently in a workload occurrence.
- ▶ Specify the maximum number of concurrent coordinator and nested activities on a database partition for a workload occurrence.
- ▶ Define the rule conditions and the actions to take when activities exceed the rule boundaries.

To select Create limits for concurrent activities from WLM Scheme by Objectives screen, expand the project tree view (WLMDB in our example), right click **Workload Management Schemes** → **New** → **Workload Management Scheme**. In the New file for Workload Management Scheme panel, give a scheme name and select **Create a scheme by objective**.

In the Workload Management Scheme by Objective panel (Figure 10-47) select **Create limits for concurrent database activities** and click **Finish**.

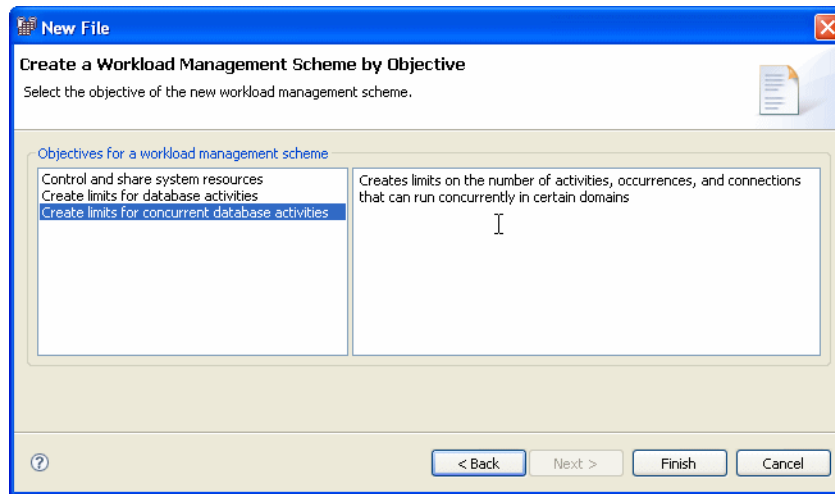


Figure 10-47 Create limits for concurrent database activities

The Create limits for the concurrent database activities main screen shown containing five tabs: General, Superclasses, Work Identities, Work Types and Create Limits. See Figure 10-48.

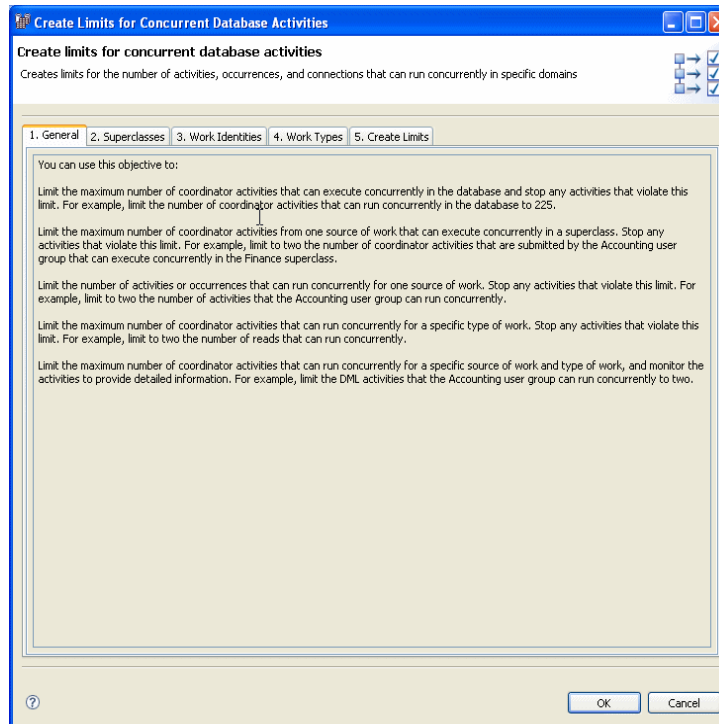


Figure 10-48 Create limits for concurrent database activities - Main screen

- ▶ **General:**
This is an informational tab. Select **Superclasses** tab to continue.
- ▶ **Superclasses, Work Identities, and Work Type sets:**
Using the same process described in “Controlling and sharing system resource” on page 248, to create superclasses, work identities, and work type sets.
- ▶ **Create Limits**
Use this tab (Figure 10-49 on page 283) to create limits for concurrent activities.

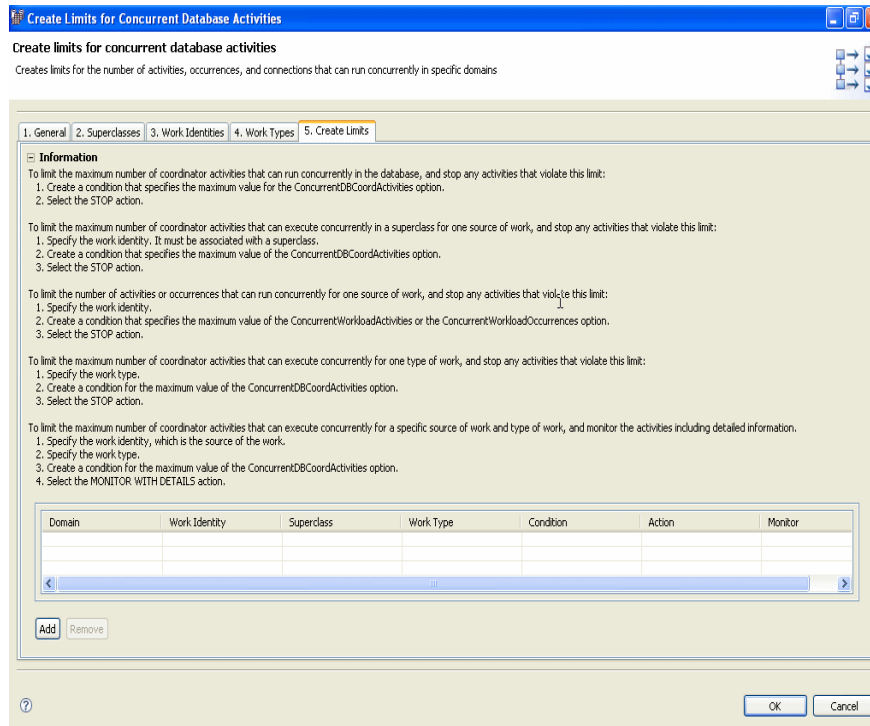


Figure 10-49 Create limits for concurrent database activities - Create limits tab

Click **Add** and choose the type of limit you want to set as shown in Figure 10-50 on page 284.

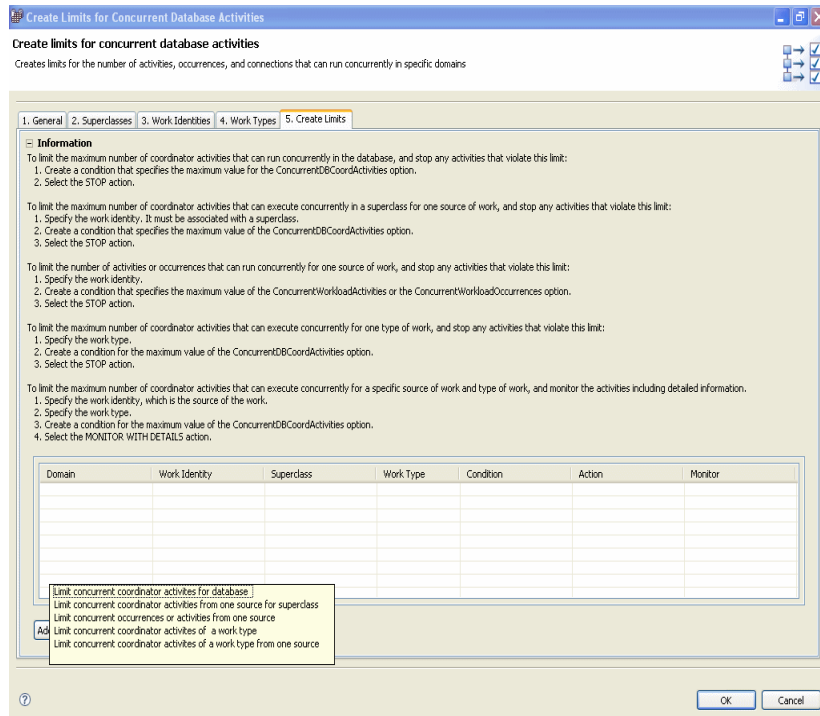


Figure 10-50 Create limits for concurrent database activities - Create limit Options

You are provided with five set options to choose from:

- Limit concurrent coordinator activities for database

In this option you can,

- Define the condition for a limit on the database domain.
- Specify type of action to be taken when the concurrency limit is exceeded.

- Limit concurrent coordinator activities from one source for superclass


In this option, you can

- Specify the work identity that is the source of the work.
- Define the condition for a limit on the superclass domain.
- Specify type of action to be taken when the concurrency limit is exceeded.

- Limit concurrent occurrences or activities from one source

In this option, you can perform the following

- Specify the work identity that is the source of the occurrence or activity.
 - Define the condition for a limit on the work identity domain.
 - Specify the action to take when the concurrency limit is exceeded.
- Limit concurrent coordinator activities of a work type
- In this option, you can perform the following
- Specify the work type that you want to limit.
 - Define the condition for a limit on the work type domain.
 - Specify with the action to take when the concurrency limit is exceeded.
- Limit concurrent coordinator activities of a work type from one source
- In this option, you can perform
- Define the condition for a limit on the subclass domain.
 - Specify the action to take when the concurrency limit is exceeded.
 - Specify the work identity and work type combination that you want to limit.

Note: The help information can be turned on or off using the twist icon  on the top left corner on the Create Limits tab.

For our example, we select **Limit concurrent coordinator activities for database** to restrict concurrent instance of an activity. An entry is added to with required fields “opened” (not greyed out). Design Studio presents a browse button next to the field when you click on the “opened” field. See Figure 10-51.

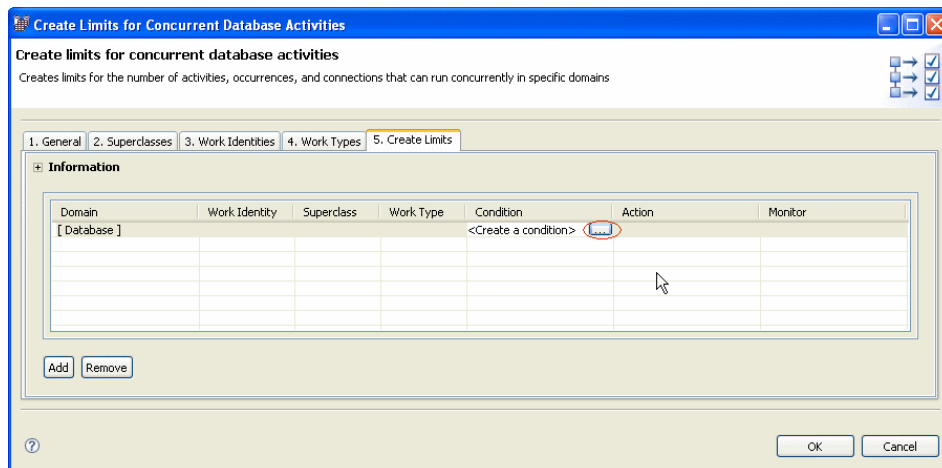
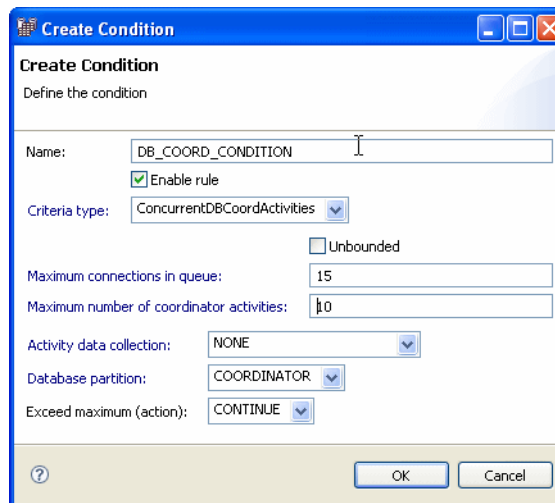


Figure 10-51 Create limits - Create a condition

In our example, the Create Condition panel presents where you can specify concurrency control configuration. See Figure 10-52. To see the description of a field, left click on the field name.



The screenshot shows a Windows-style dialog box titled "Create Condition". The main area is labeled "Define the condition". It contains several input fields and checkboxes. The "Name" field is filled with "DB_COORD_CONDITION". The "Enable rule" checkbox is checked. The "Criteria type" dropdown is set to "ConcurrentDBCoordActivities". There is an unchecked "Unbounded" checkbox. The "Maximum connections in queue" field contains the number "15". The "Maximum number of coordinator activities" field contains the number "10". The "Activity data collection" dropdown is set to "NONE". The "Database partition" dropdown is set to "COORDINATOR". The "Exceed maximum (action)" dropdown is set to "CONTINUE". At the bottom right, there are "OK" and "Cancel" buttons, and a help icon (?) on the bottom left.

Figure 10-52 Concurrency Control - Create new control

Note: Setting the value for the maximum number of connections allowed in a queue to unbounded is not recommended. There might be problems if you have limited the number of connections allowed for the database. In this case, unbounded queues let the queued activities to use all of the allowed connections so that unrelated but legitimate work might be locked out unexpectedly.

After completing the Create Condition screen, click **OK**. This creates the control rule for concurrency control for database activities as shown in Figure 10-53.

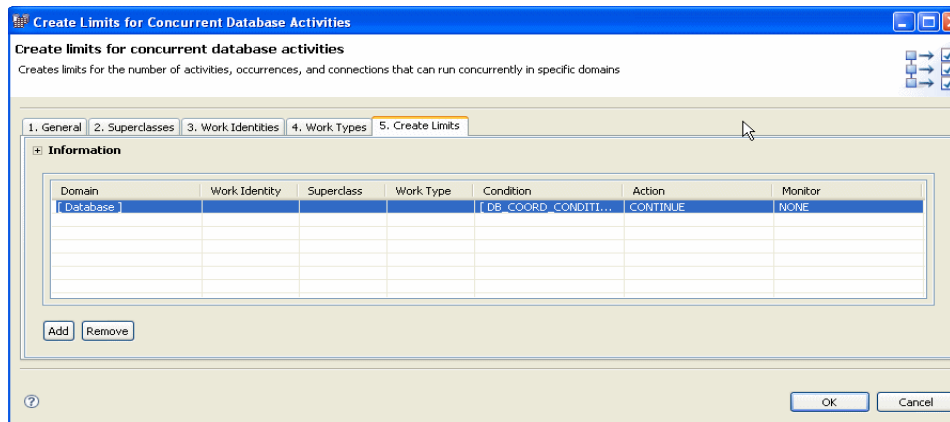


Figure 10-53 Create limits for Concurrent database activities showing Control Rule

Additional control rules can be added using **Add** button. After all the control rules are defined, click **OK**. Design Studio creates a WLM Scheme and take you back to the default Business Intelligence (BI) perspective. The BI Perspective is expand to show the final output, see Figure 10-54.

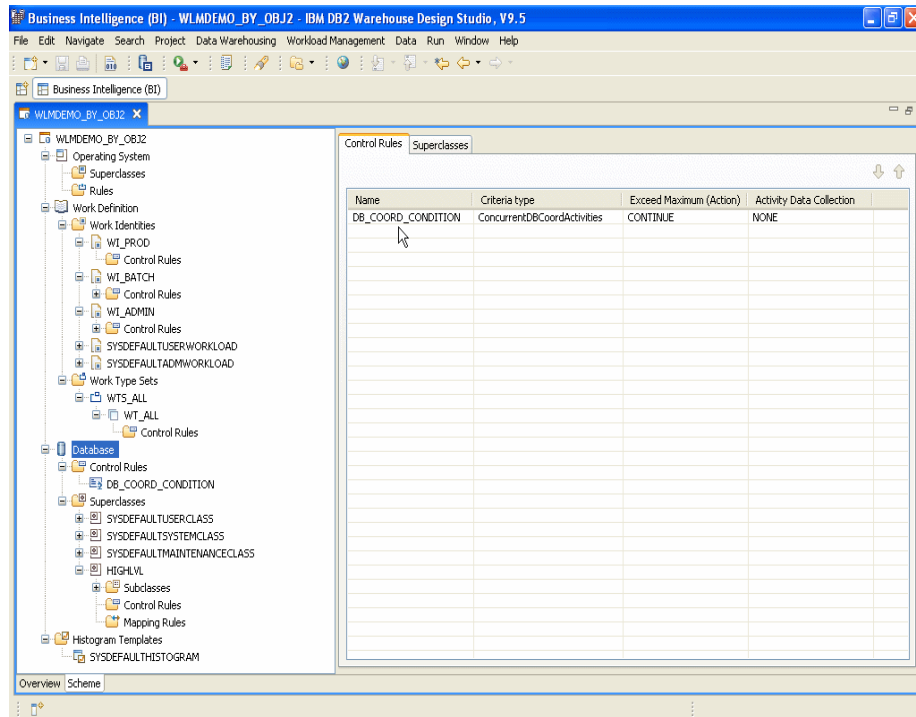


Figure 10-54 Control Rule to limit concurrent coordinator activities for database - Tree view

Validating WLM scheme and generating code

Once the workload scheme is created, you can use the Design Studio Validate option to validate the workload, service classes, and the relationship you just created. After successful validation you can generate code using Generate Code feature. For the details, refer to “Validating” on page 261 and “Generating code” on page 262.

10.3.2 Create workload scheme by yourself

You can use this workload creating options to create a WLM scheme and work with entities in the scheme from scratch. This method does not guide you to create work identities, superclass, subclass, work type and work type sets. You need to create everything by yourself using the scheme tree view. You can create a new project for the new WLM scheme or create the new scheme under an existing project.

To create a new WLM scheme using this method, select **Create a scheme yourself** and enter WLM scheme name as shown in Figure 10-55. In this example, we create a WLM scheme WLMDEMO_BY_YRSLF.

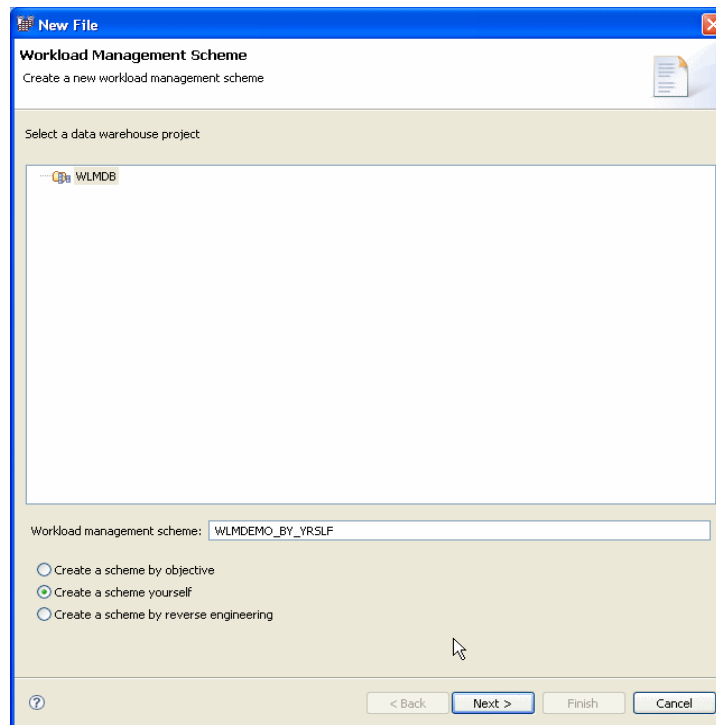


Figure 10-55 Create scheme yourself

Click **Next** and the Workload Management Scheme Options panel shown for you to define a work action set name. By default, it takes the WLM scheme name you provided as the work action set name. We leave the default work action set name and click **Finish**. The file name created is the scheme name with *.wlms* under the default workspace defined.

In addition to the default work identities, DesignStudio also shows you these defaults:

- ▶ Default super classes (SYSDEFAULTSYSTEMCLASS, SYSDEFAULTUSERCLASS)
- ▶ Default sub class under every super class (SYSDEFAULTSUBCLASS)
- ▶ Default histogram (SYSDEFAULTHISTOGRAM)

Design Studio creates the WLM scheme with only the default work identities as shown in Figure 10-56 on page 290. So, you have to create all other user-defined entities such as superclass, subclass, work identities, work type, work type sets, control rules and mapping rules using the tree view.

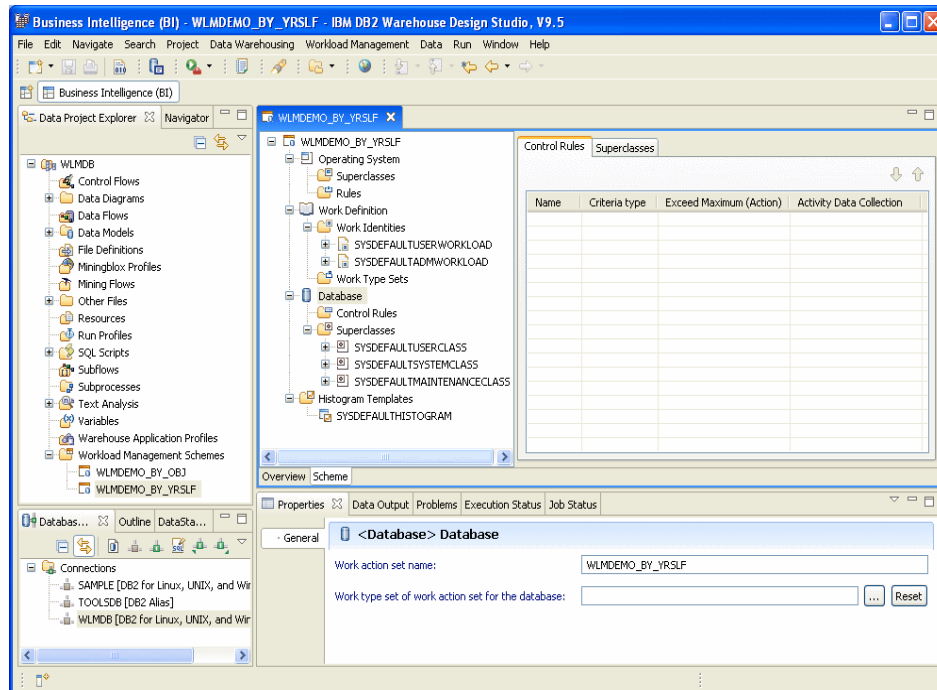


Figure 10-56 WLM Scheme created using create yourself option

To create a new entity, right click on the object and select the option. We show here the steps to create a new superclass. For example, right-click **Superclasses** and select **New Superclass** as shown in Figure 10-57.

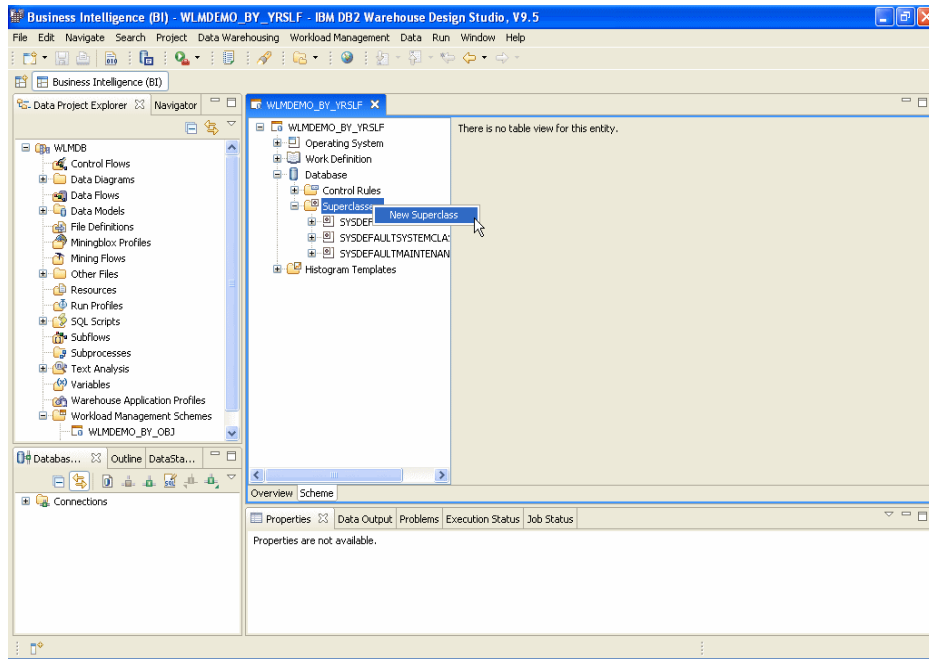


Figure 10-57 Create Scheme by Yourself - Create New Superclass

You are required to enter an unique name in the New Superclass window and provide additional info in the Properties view (Figure 10-58).

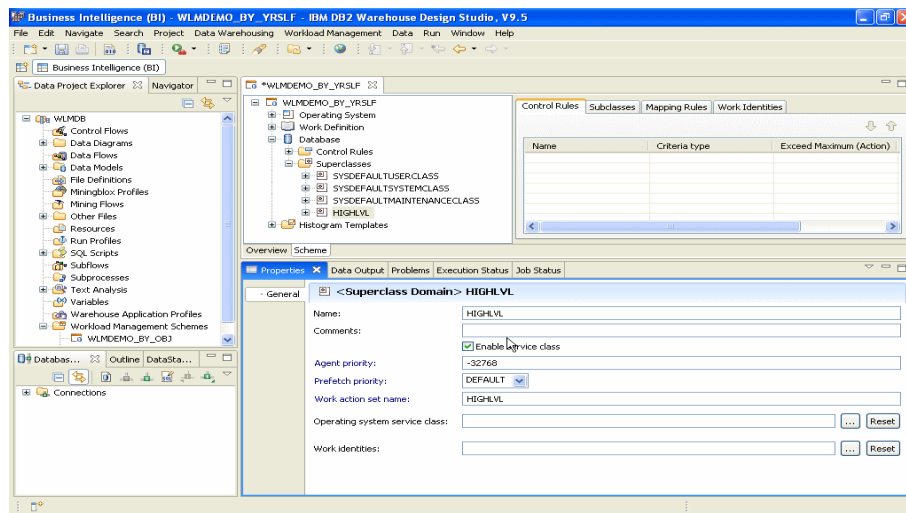


Figure 10-58 Create by Yourself - New Service Class properties view

After creation of all the superclasses, subclasses, work identities, work type sets and work types, control rules and mapping rules, validate the scheme by **Workload Management** → **Validate**. Use the Generate Code menu option to generate DB2 statements for deployment.

10.3.3 Create workload scheme by reverse engineering

This option allows you to extract the information about an existing WLM scheme from DB2 database to Design Studio as. You can then modify and add to the new scheme to make it unique.

When reverse engineering, Design Studio extracts the settings and entities that make up a workload scheme from database and create a new WLM scheme. Table 10-2 shows the DB2 objects that map to the Design Studio WLM entities.

Table 10-2 Mapping between DB2 objects and Design Studio entities

DB2 WLM objects	Design Studio WLM entities
Service superclass	Superclass
Service subclass	Subclass
Workload	Work identity
Work class set	Work type set
Work class	Work type

DB2 WLM threshold rules and enforcement criteria such as work action sets and corresponding to Design Studio control and mapping rules are shown in Table 10-3.

Table 10-3 Control rules mapping

DB2 WLM control criteria	Design Studio WLM rule
Threshold for the database domain	Control rule for the database
Threshold for the superclass domain	Control rule for a superclass
Threshold for the subclass domain	Control rule for a subclass
Threshold for the workload domain	Control rule for work entity
WHEN ACTION in a WORK ACTION SET for the database	Control rule for a work type (WHEN)
MAPPING ACTION in a WORK ACTION SET for service superclass	Mapping rule for a superclass (MAP ACTIVITY)

Reverse engineering steps

The Design Studio workload management reverse engineer steps are:

1. In the Business Intelligence perspective, select **File** → **New** → **Workload Management Scheme**.
2. In the Workload Management Scheme panel (Figure 10-59), type a unique name in the Workload management scheme field, select **Create a scheme by reverse engineering**, and click **Next**.

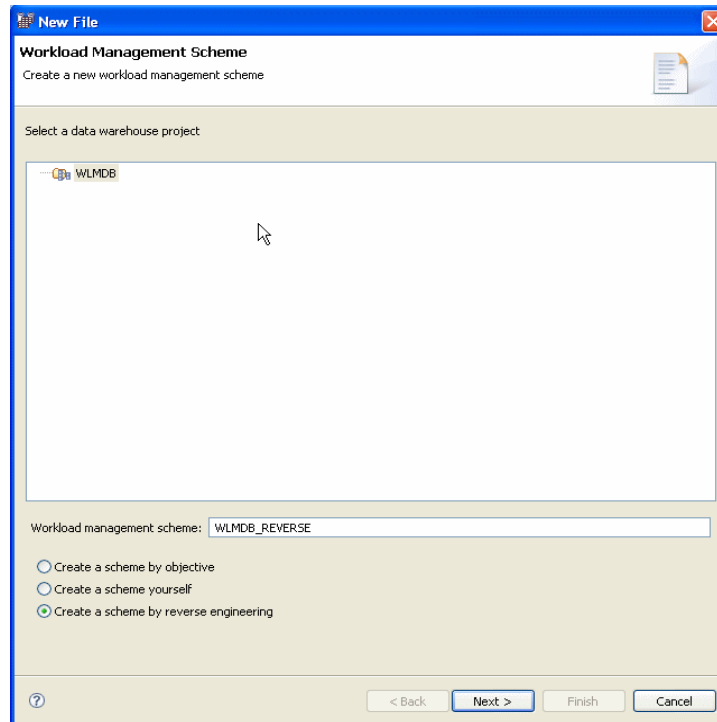


Figure 10-59 Workload Management Scheme - Reverse Engineering a Scheme

3. In the Select Connection panel (Figure 10-60 on page 294), connect to the database where the collection of information about the scheme exists using the existing connection or create a new connection. We select **Create a new connection**.

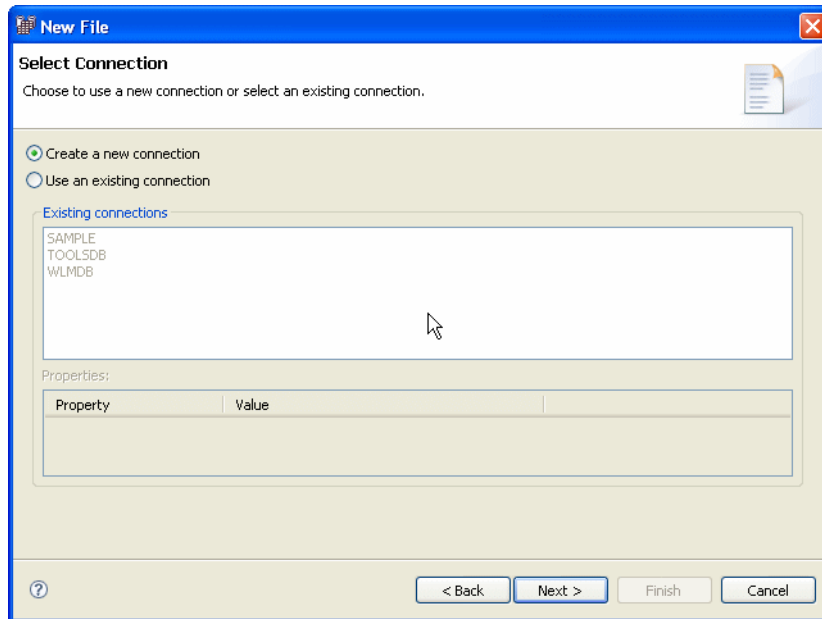


Figure 10-60 Create a new database connection

4. Complete the Connections Parameters (Figure 10-61 on page 295) panel, and click **Finish**.

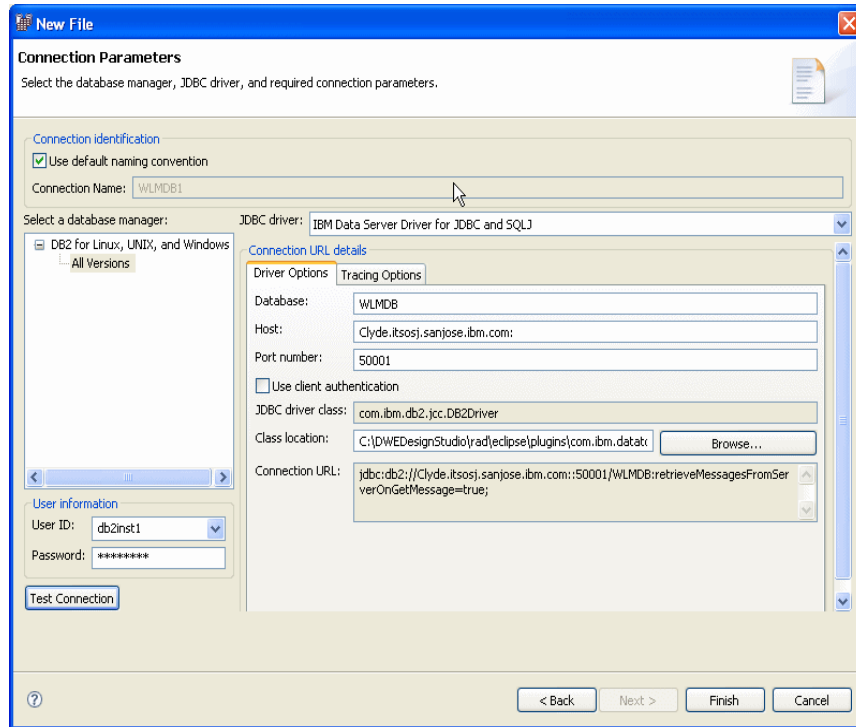


Figure 10-61 New Database Connection parameters

5. Design Studio retrieves the information about the scheme from the database, creates the new scheme, and creates a file for the scheme. The file named is <WLM scheme name>.wlm. The output is shown in Figure 10-62 on page 296. Now, you can modify the scheme according to your specifications.

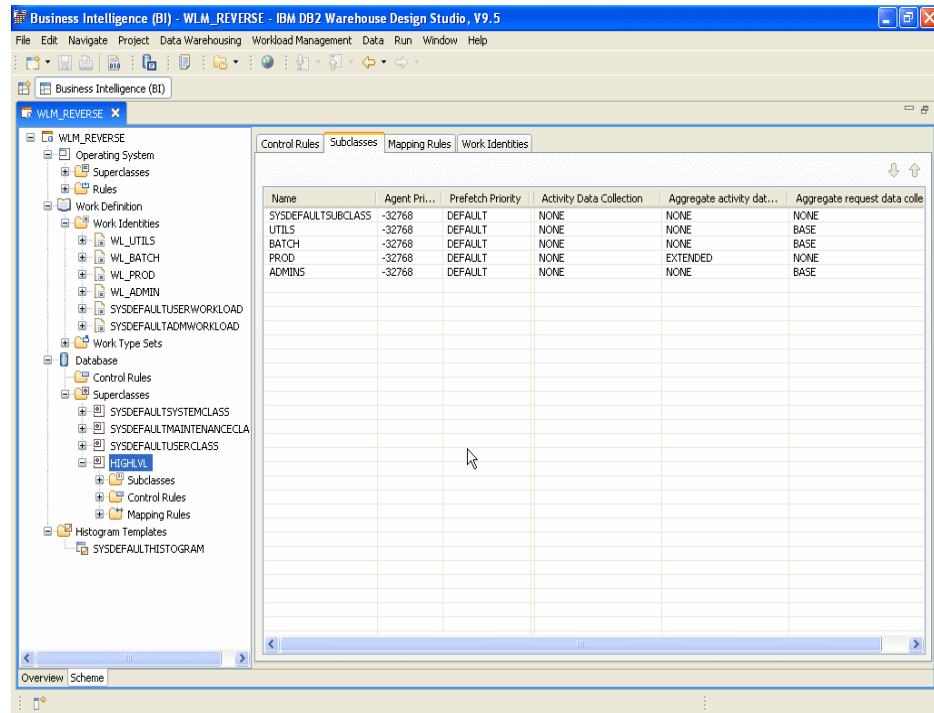


Figure 10-62 WLM Schema created from Reverse Engineering process

10.4 Execute a workload management scheme

You can execute a workload management scheme directly from Design Studio to the target database. You can do clean or delta execution without first using the Generate Code function.

To generate SQL code, perform the following steps:

1. In the Data Project Explorer, expand the data warehouse project that contains the scheme that you want to work with, and then expand the Workload Management Schemes folder.
2. Select the scheme that you want to work with, select **Workload Management** → **Validate**
3. Select **Workload Management** → **Generate Code**

If you get errors or warnings, open the Problems view to learn more about the problems description, and use Help topics to learn how to correct the scheme.

If you incur error and warning messages, correct the definitions as required and then repeat code generation procedure.

After the code generation process succeeds, you are ready to deploy the scheme.

Note: During **Workload Management** → **Generate Code**, if the database does not contain any WLM Scheme, Design Studio will show information screen stating “No code was generated because the scheme is empty”.

Design Studio provides two approaches to run the WLM scheme, Execute and Delta Execute.

Execution

This approach is a *clean* execution. When you execute workload management scheme, it begins with wiping off all the existing WLM configuration on the target and create the entire scheme fresh in the target. If the execution fails, the settings for the WLM configuration that exist in the database are lost.

We recommend use this option only when you like to create have a fresh start.

To execute a workload management scheme:

1. With the workload management scheme open, select **Workload Management** → **Execute**.
2. In the Generated Code window, review the code that will be executed in the database.
3. Select **Execute** in database and click **Next**.
4. In the Execution Options window, specify any options and click **Next**.
5. In the Select Connection window, connect to the target database.
 - You can select **Create a new connection**, click **Next**, complete the Connections Parameters window, and click **Finish**.
 - You can select **Use an existing connection**, select that connection, and click **Finish**.
6. In the Execution Result panel, review the execution log information. You can also save the execution log information.

Figure 10-63 shows the execution results panel using Execute.

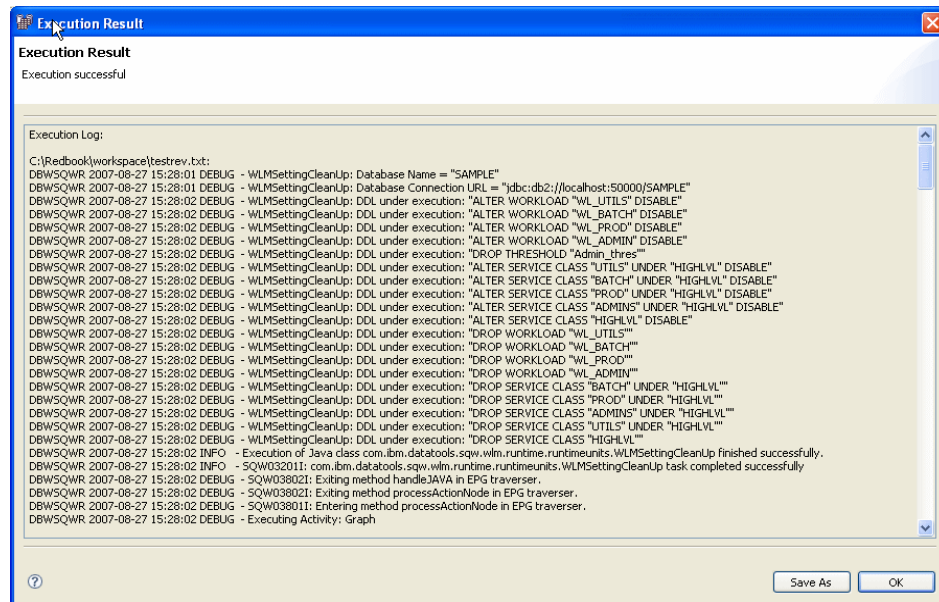


Figure 10-63 Workload Management Execute - Execution Result screen

Delta Execution

Delta Execution performs ALTER whenever possible and perform CREATE or DROP of WLM entities only when necessary. It performs reverse engineer on the target database and determine the changes.

You can use the Delta Execute to alter the database or another workload management scheme to make it identical to the current workload management scheme.

Using Delta Execute you can apply scheme settings to the database. Some of the advantages of using Delta Execute are

- ▶ Each WLM statements has been committed

Only one uncommitted WLM-exclusive SQL statement at a time is allowed across all partitions. If an uncommitted WLM-exclusive SQL statement is executing, subsequent WLM-exclusive SQL statements will wait until the current WLM-exclusive SQL statement commits or rolls back. If any error happens during Delta Execution, Design Studio will not make any changes in the database. Your database will be in the state before the Delta Execution.

- ▶ Many dependencies between WLM entities

On a complex system with many WLM entities, proper order of execution plays a critical role. Delta Execution figures out a proper order for you.

- ▶ Cannot drop WLM entities that are in use

Delta Execution avoids DROP/CREATE of WLM entities where possible. When executing WLM scheme on database, Design Studio does ALTER to match the settings in the workload management scheme, and only does DROP/CREATE if necessary.

We recommend Delta Execution for executing WLM entities since the result is the same as the result of using the Execute menu option. However when you use the Delta Execute option, the Design Studio makes only the minimal number of changes that are required to update the settings of either the database or the other workload management scheme to be identical to the current workload management scheme.

Delta Execute against database

Use the following steps to perform delta execution on a database:

1. In the workload management scheme open, select **Workload Management** → **Delta Execute**.
2. In the Comparison Options window (Figure 10-64), select **Database**, click **Next**.

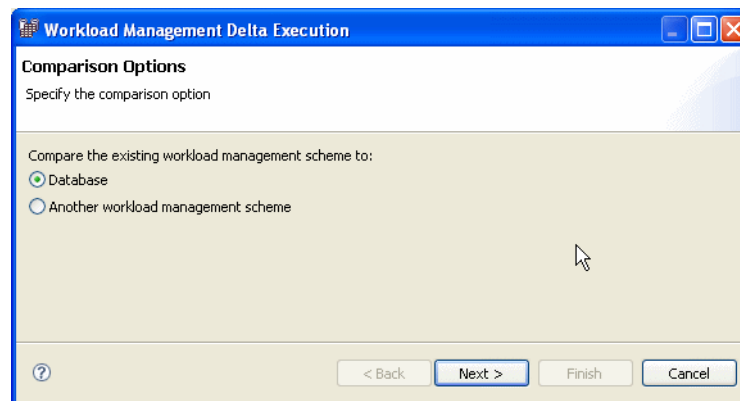


Figure 10-64 Delta Execution - Compare options

3. In the Select Connection window, select **Create a new connection** or **Use an existing connection**.
4. Delta execution compare the current WLM scheme with WLM entities in the database and generates a set of SQL statements. See Figure 10-65.

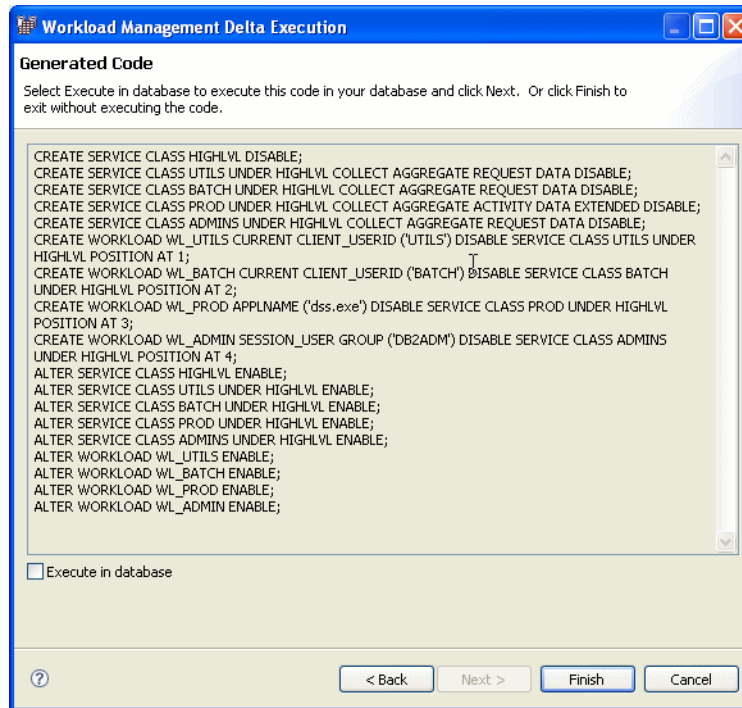


Figure 10-65 Delta execution - Generated Code

5. In Generated Code window, select **Execute in database** and click **Next**.
6. In the Execution Options window (Figure 10-66 on page 301), specify any options and click **Next**.

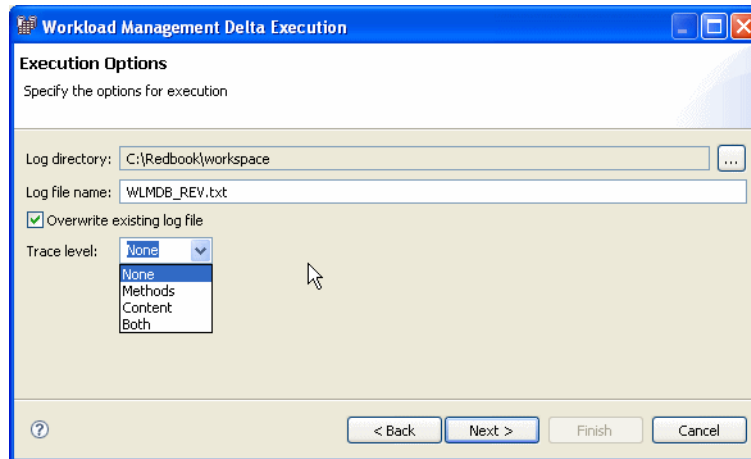


Figure 10-66 Delta Execution - Execute options screen

- In the Execution Result window (Figure 10-67), verify the result and click **Finish**.

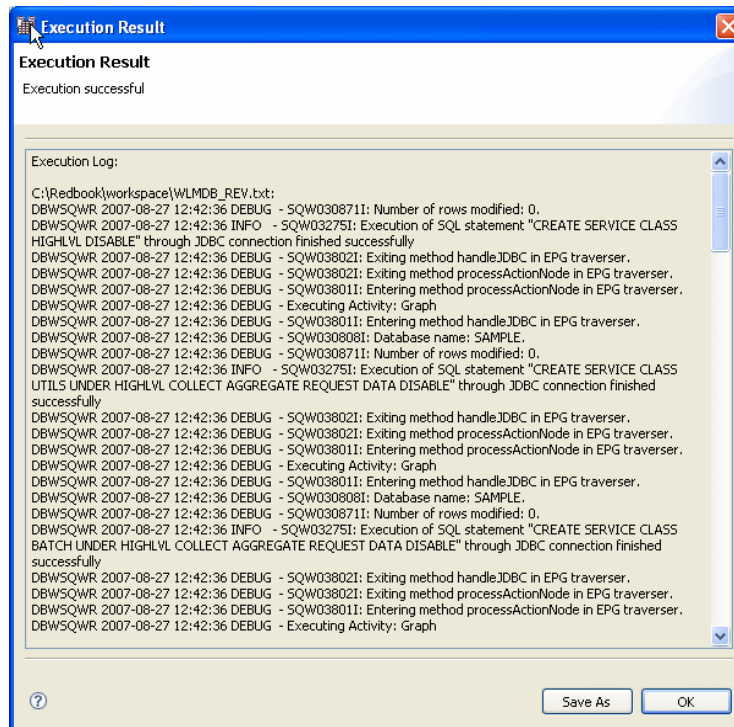


Figure 10-67 Delta Execution - Results screen

Delta Execute against WLM schemes

Using Delta Execution to compare two WLM scheme. This can be useful to fine out the WLM changes made on the database if different WLM scheme versions are available.

The steps To use the Delta Execute option,

1. In the workload management scheme open, select **Workload Management** → **Delta Execute**.
2. In the Comparison Options window, select **Another workload management scheme**, click **Next**. See Figure 10-68.

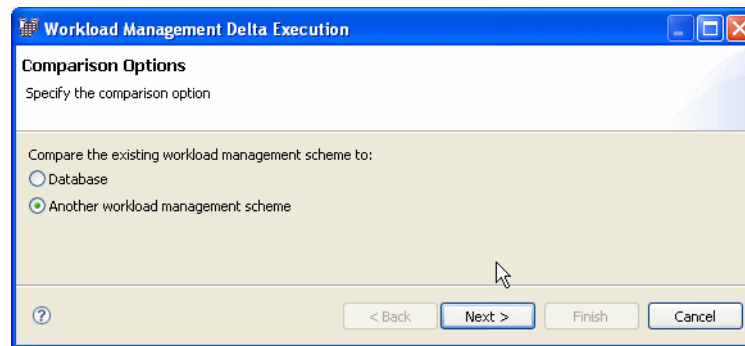


Figure 10-68 Delta Execution - Another workload management scheme

3. Choose the .wlm files of the WLM scheme to be compared with. See Figure 10-69.

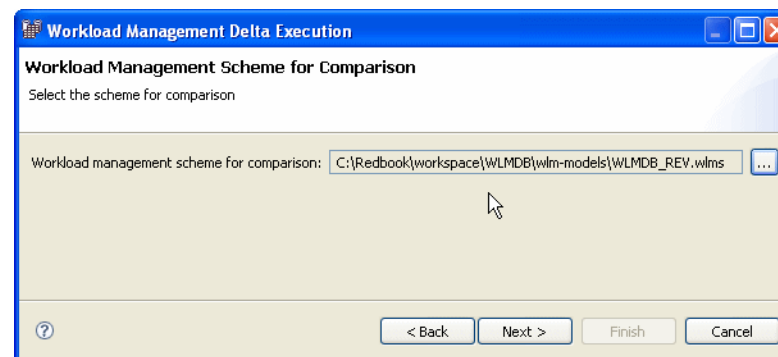


Figure 10-69 Delta Execute - Select WLM Scheme for comparison

4. The delta SQL statement is shown in the Generated Code window (Figure 10-70).

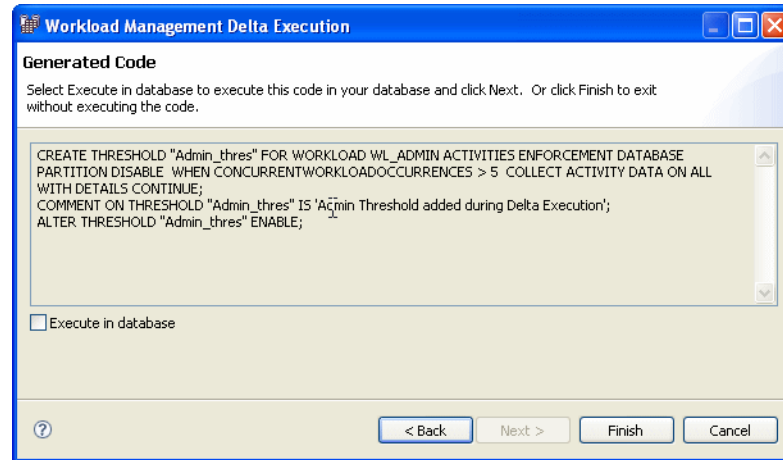


Figure 10-70 Delta Execute - Compare WLM Schemes results

10.5 AIX WLM management

DB2 WLM is integrated with AIX Workload Manager. You can use the Design Studio graphical user interface to set up and maintain both AIX WML and DB2 WLM service classes. In DB2 9.5, you can utilize the AIX WLM to manage only the CPU utilization.

10.5.1 Creating operating system service classes and limits

Using Design Studio, you can create operating system service classes when a new WLM schemes is created or by modifying the existing WLM scheme using the tree view.

Creating OS service classes on an existing DB2 WLM scheme

Before you begin using Design Studio to create AIX WLM service classes, it is assume that you have the following created:

- ▶ A WLM scheme
- ▶ DB2 work identities, superclasses, and subclasses

Plan mapping between DB2 and AIX WLM mapping scheme, for example, 1:1 or flat mapping. We recommend to start with 1:1 mapping and then expand.

Create operating system superclass

The steps to create the AIX WLM superclasses are

1. In the Data Project Explorer, expand the data warehouse project that contains the scheme that you want to work with
2. Expand the Workload Management Schemes folder.
3. Double-click the scheme for which you are creating an operating system service class and expand the scheme.
4. Right-click **Operating System**, select **New Operating System Superclass**.
5. In the New Operating System Service Class window (Figure 10-71), type a unique name and click **OK**.

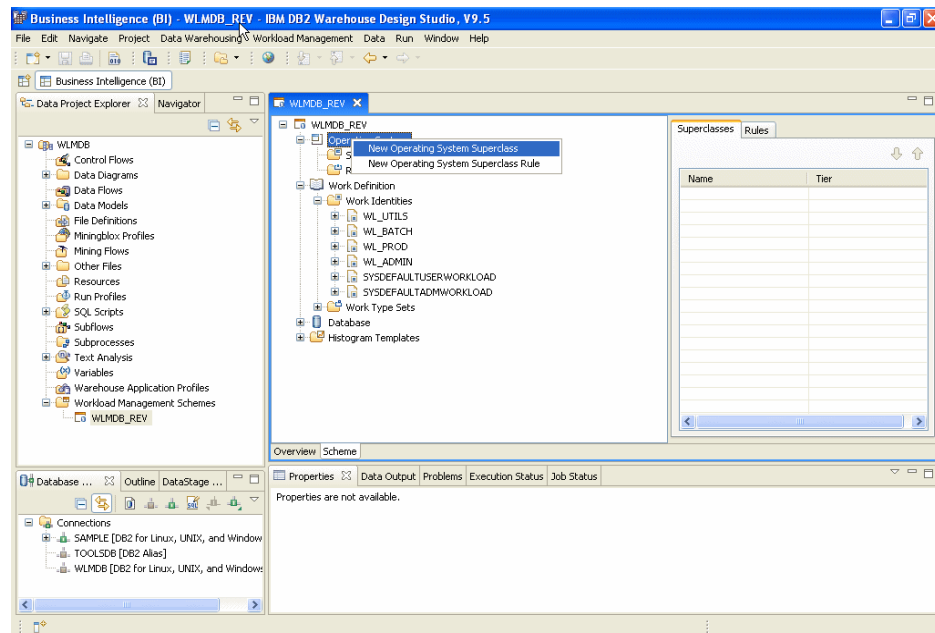


Figure 10-71 Create new operating system superclass

- ▶ Define the properties of the operating system superclass by completing the General tab of the Properties view as shown in Figure 10-72.

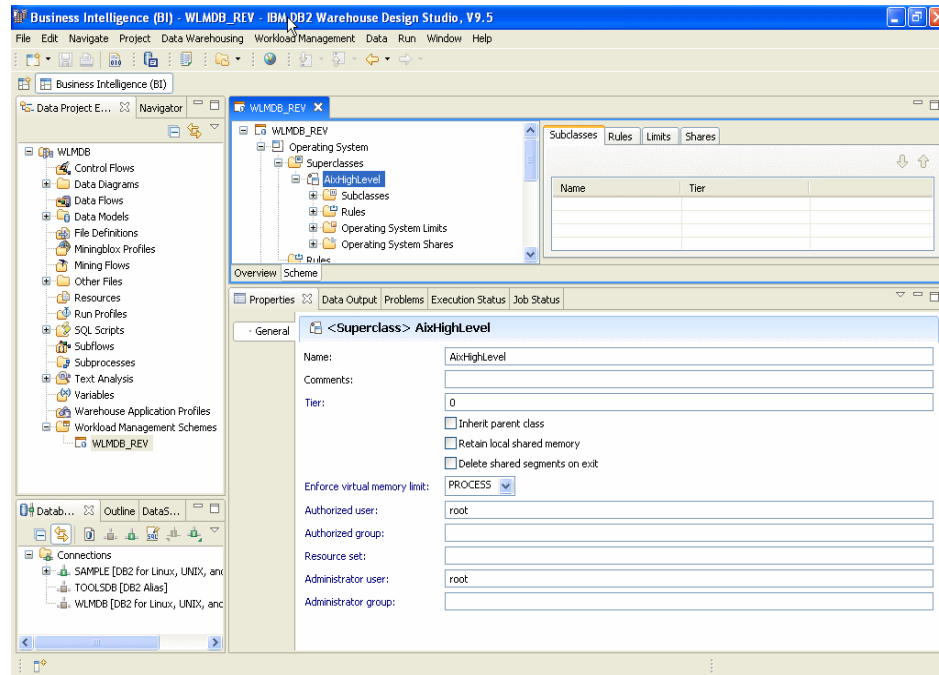


Figure 10-72 Operating system superclass properties

Note: DB2 service classes cannot work with the AIX Workload Manager inheritance feature. In Design Studio, inheritance attribute was not enabled by default. If inheritance is enabled, the DB2 workload manager cannot change the AIX workload management class of a thread using tagging. This restriction makes any integration of DB2 Workload Manager and AIX Workload Manager unusable. The DB2 data server cannot detect whether AIX Workload Manager inheritance is enabled and does not issue an error message if inheritance is enabled.

Create operating system subclass

If necessary, define the operating system subclasses under the superclass. For subclass, you can define any of the following:

- ▶ Process assignment rules.
- ▶ Resource usage limits.
- ▶ Resource shares.

The steps to create an operating system subclass are

1. In the Data Project Explorer, expand the data warehouse project that contains the scheme that you want to work with, and then expand the **Workload Management Schemes** folder.
2. Double-click the scheme for which you are creating an operating system subclass and Expand the scheme in the Scheme view.
3. Expand the Operating system and then the superclasses folder.
4. Right-click the appropriate superclass and select **New Operating System Subclass**. See Figure 10-73.

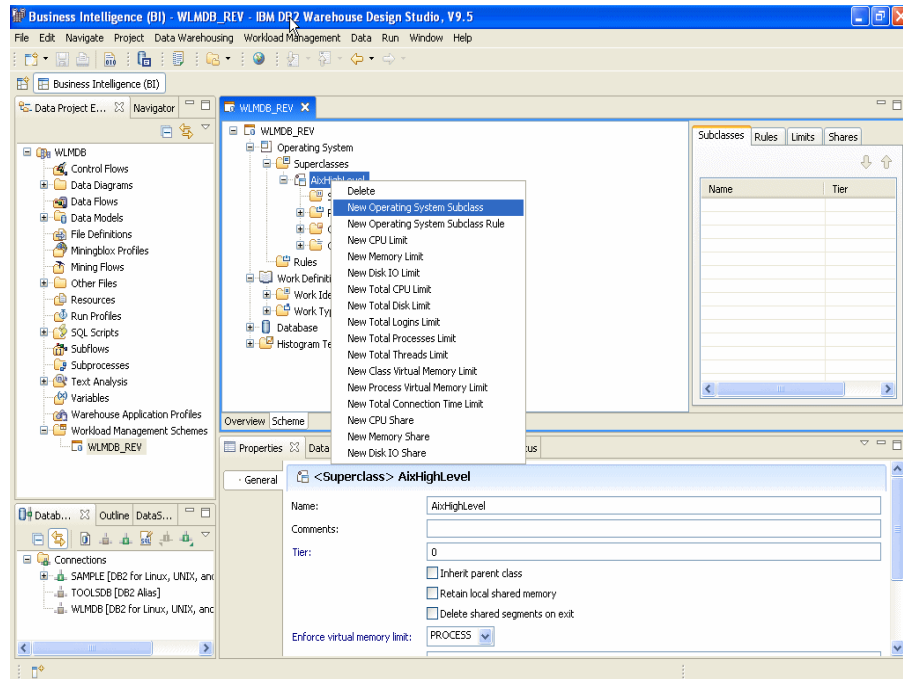


Figure 10-73 Create new operating system subclass

5. In the New Operating System Subclass window, type a unique name and click **OK**.
6. Define the properties of the operating system subclass by completing the General tab of the Properties view. See Figure 10-74.

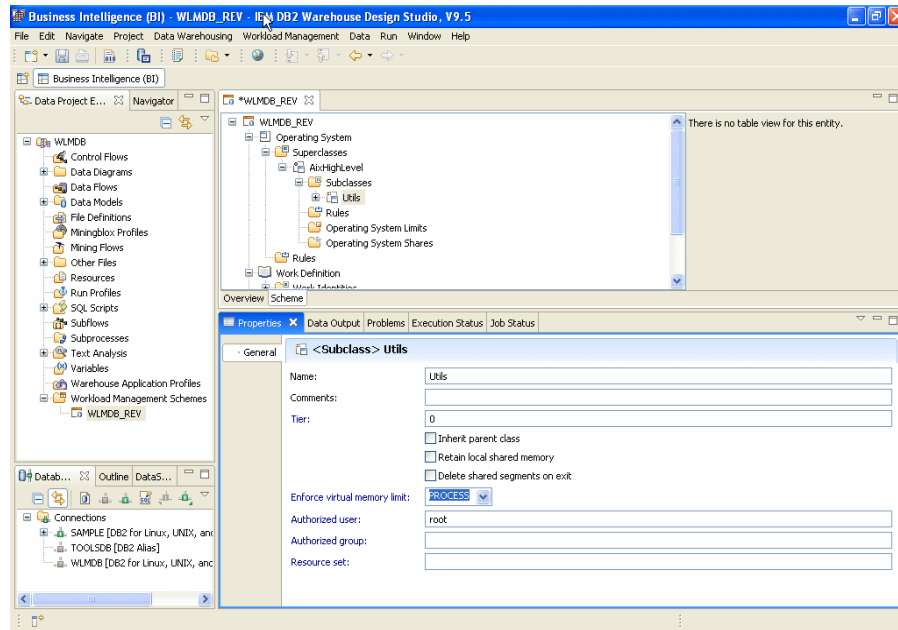


Figure 10-74 Operating system subclass - Properties view

Limiting system resources for OS service classes:

The steps to create a limit for a system resource are

1. In the Data Project Explorer, expand the data warehouse project that contains the scheme that you want to work with, and then expand the Workload Management Schemes folder.
2. Double-click the scheme for which you are creating an assignment rule.
3. In the Scheme view, expand the scheme, Operating System, Superclasses, and the appropriate superclass. right-click **Operating System Limits** at the superclass level and select the system resource. See Figure 10-75 on page 308.

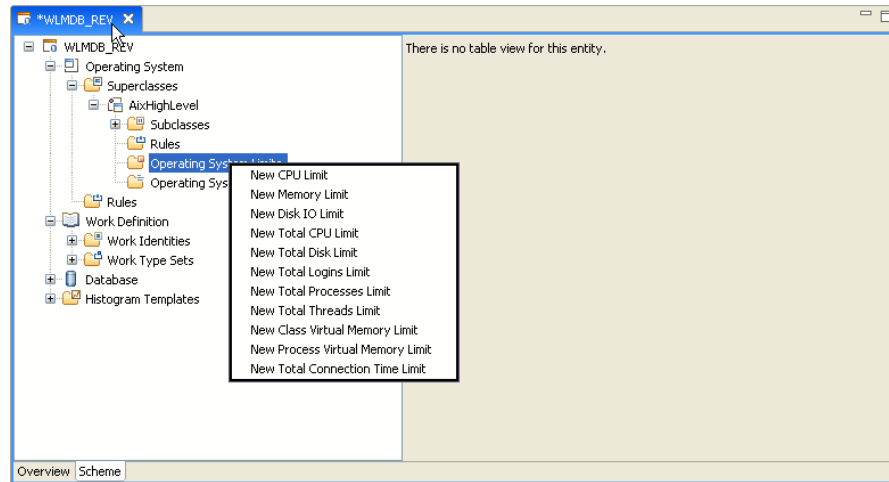


Figure 10-75 Operating system limits - superclass

You also can create the operating system limits on subclass level by selecting the resource on the subclass level as shown in Figure 10-76.

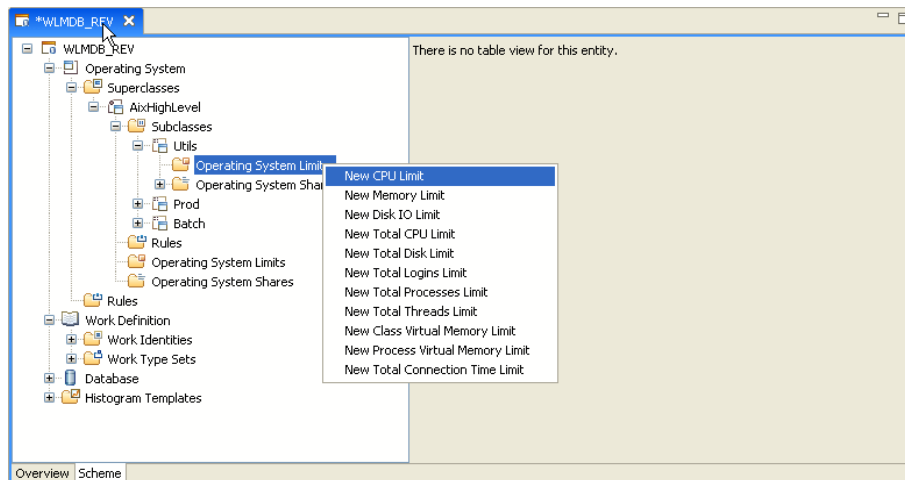


Figure 10-76 Create operating system limits - subclass

4. In the prompted window, enter a unique name for the limit and click **OK**.
5. Define the properties of the limit by completing the General tab of the Properties view, see Figure 10-77 on page 309.

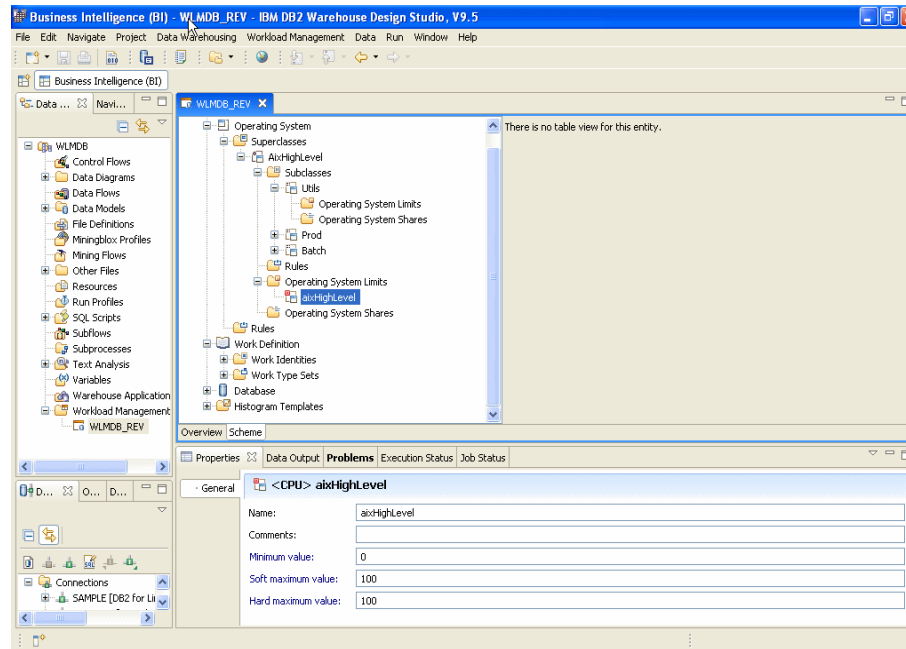


Figure 10-77 Create Operating system limits - Properties view

Allocating resource shares to OS service classes

The steps to create a share of a system resource are:

1. In the Data Project Explorer, expand the data warehouse project that contains the scheme that you want to work with, and then expand the Workload Management Schemes folder.
2. Double-click the scheme for which you are creating an assignment rule.
3. In the Scheme view, expand the scheme, Operating System, Superclasses, and the appropriate superclass. right-click **Operating System Shares** at the superclass level and select the system resource. See Figure 10-78 on page 310.

You also can create operating system share on the subclass level by selecting Operating System Shares at subclass level.

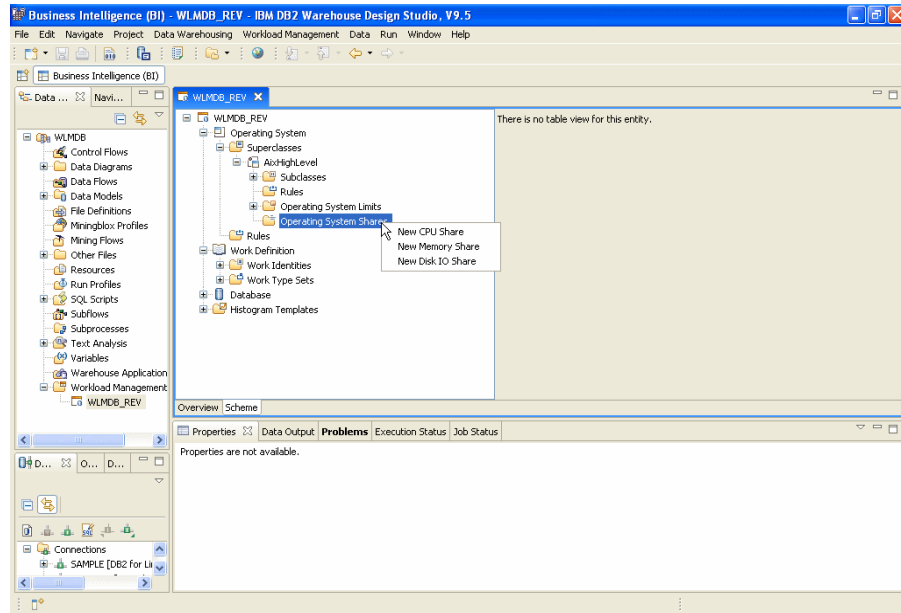


Figure 10-78 Create operating system share - Superclass

4. In the window, type a unique name for the share and click **OK**.
5. Define the properties of the system resource share by completing the General tab of the Properties view. See Figure 10-79 on page 311.

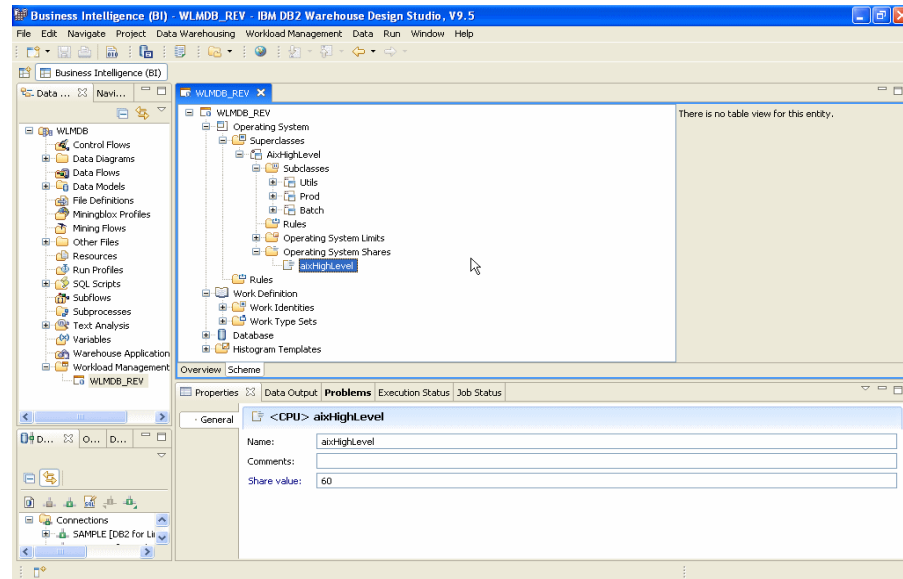


Figure 10-79 Create operating system share - Superclass Properties view

10.5.2 Configure AIX WLM using Design Studio

After the operating system superclasses, subclasses, rules, limits, and shares are created, you can associate the DB2 service classes with the operating system service classes.

In DB2 9.5, only CPU allocation control is supported when integrating the AIX Workload Manager with DB2 workload management. The other parameters such as memory and I/O settings are not supported. DB2 database-level memory is shared among all agents from different DB2 service classes, therefore, you cannot divide memory allocation between different service classes. To control I/O, you can use the prefetcher priority attribute of a DB2 service class to differentiate I/O priorities between different DB2 service classes.

Associate DB2 superclass with AIX superclass

The steps to associate DB2 superclass with AIX superclass are

1. In the Scheme view, expand the scheme, Database, Superclasses and select the superclass you like to associate with the operating system superclass.
 - ▶ In the General tab of the Properties view for DB2 superclass, associate with AIX service class, by selecting the button for Operating system service class. Figure 10-80 on page 312 shows an example of associating AIX superclass with DB2 superclass.

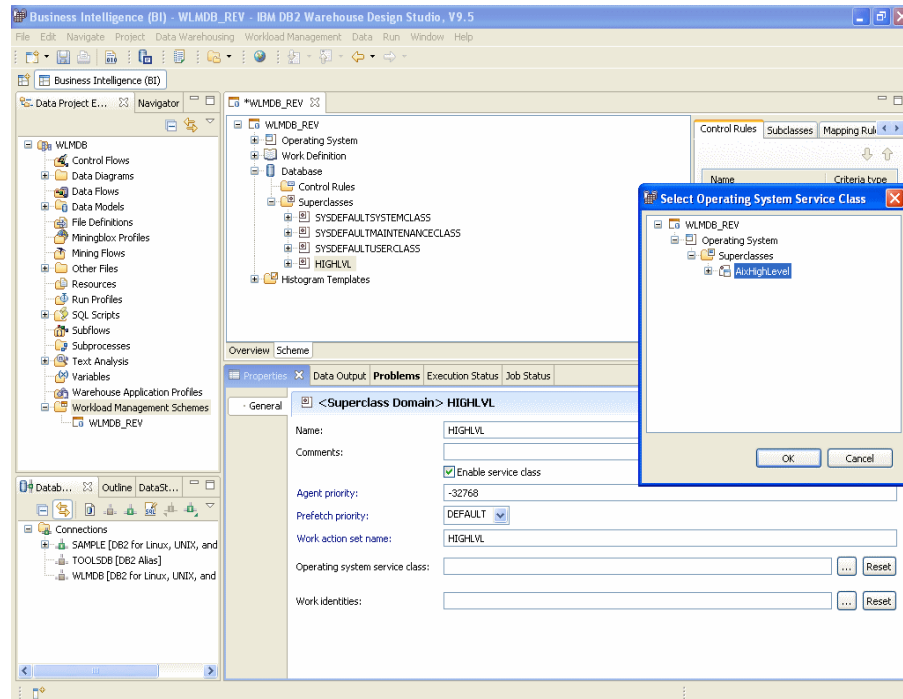


Figure 10-80 AIX superclass and DB2 superclass association

Adding control rules to AIX service classes

Once the service classes association is completed, you can define the control rules on AIX service classes to manage the resource.

The steps to adding the rules are

1. In the Scheme view, expand the scheme and Operating System, right click **Rules** at the superclass level and select **New Operating System Superclass Rule**. See Figure 10-81 on page 313.

To add rules for subclass, expand the view to subclass level and right click **Rules** at the subclass level.

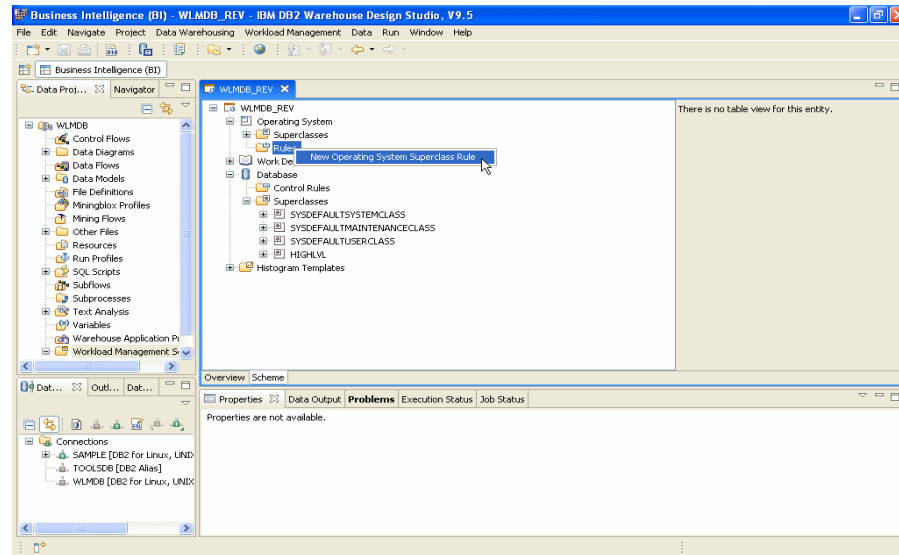



Figure 10-81 Creating new operating system superclass rule

2. In New Operating System Superclass Rule, type a unique name for the rule and click **OK**.
3. Define the properties of the rule by completing the General tab of the Properties view.
4. To associate Operating system superclass, select the  button for Operating system service class and select the superclass you want.
5. Figure 10-82 shows that we created aixHighLvl_rule and associate it with Operating system superclass AIXHighLevel.

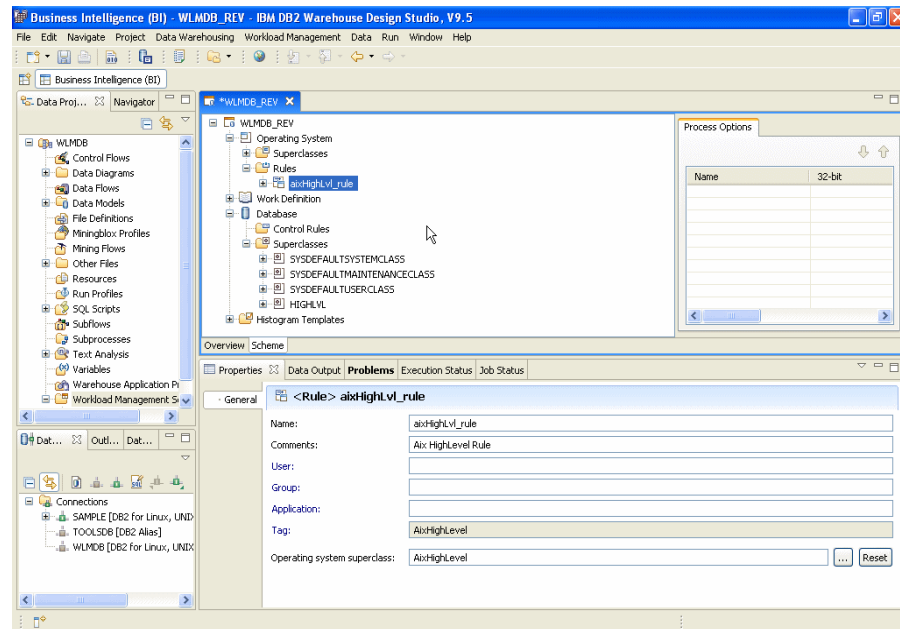


Figure 10-82 Operating system rule for superclass

Note: In Design Studio, Application Tag is a read-only field. Design Studio generate a string value for OUTBOUND CORRELATOR property that DB2 uses.

Validate and Generate Code

After creating DB2 service classes and AIX service classes, associating them, and creating rules, you can validate what you have created using Validate and Generate Code function.

Figure 10-83 shows the confirmation screen from Generate code execution.

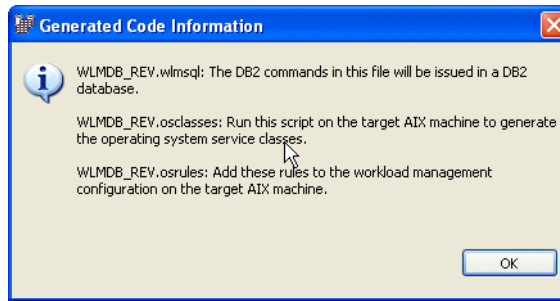


Figure 10-83 Generate code information screen

The Generated Code Information window lists the files that are created and their usage. Click **OK** and Design studio automatically opens the files generated in the Editors view. Any of three possible files that can be generated are:

- ▶ DB2 command code file

This file is generated and listed in the message when DB2 workload management entities exist in the scheme. The file is created with Scheme name and *.wlmsql* extension. For example, WLMDB_REV.wlmsql.

Figure 10-84 shows the DB2 SQL generated by Design Studio.

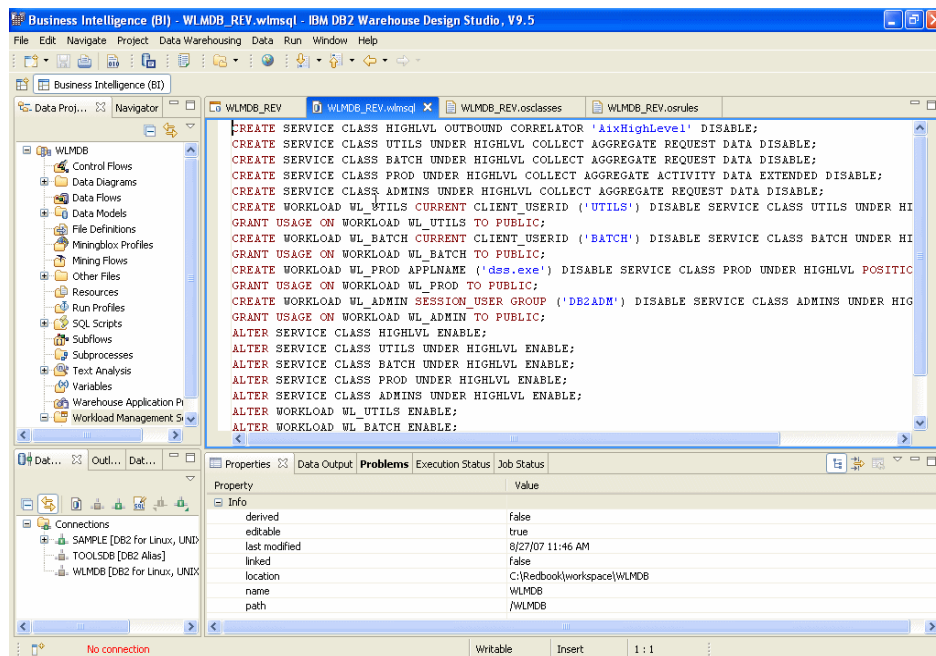
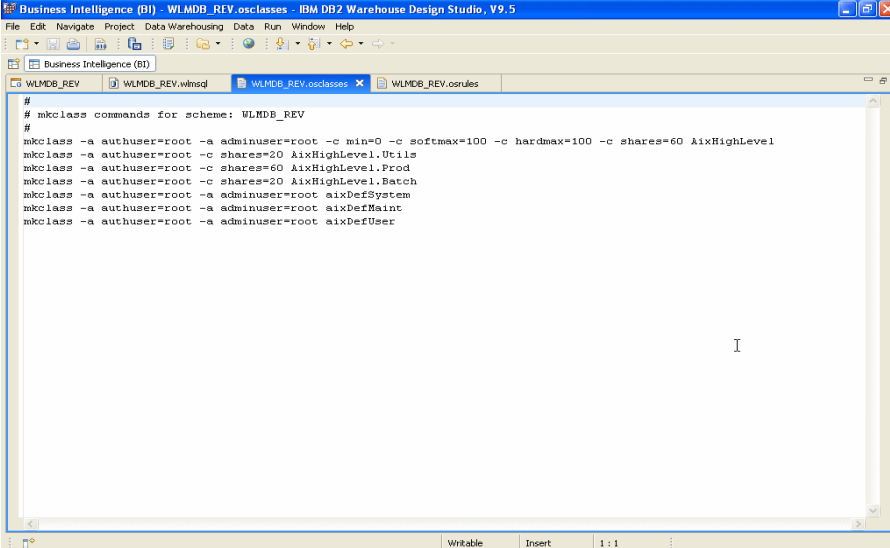


Figure 10-84 Generated SQL code from Design Studio

► **Classes file**

This file is generated and listed when operating system service classes exist in the scheme. The file is created with scheme name and *.osclasses* extension, for example, *WLMDB_REV.osclasses*.

Figure 10-85 shows the sample output of operating system class file.



```

Business Intelligence (BI) - WLMDB_REV.osclasses - IBM DB2 Warehouse Design Studio, V9.5
File Edit Navigate Project Data Warehousing Data Run Window Help
Business Intelligence (BI)
WLMDB_REV WLMDB_REV.wmoad WLMDB_REV.osclasses WLMDB_REV.osrules

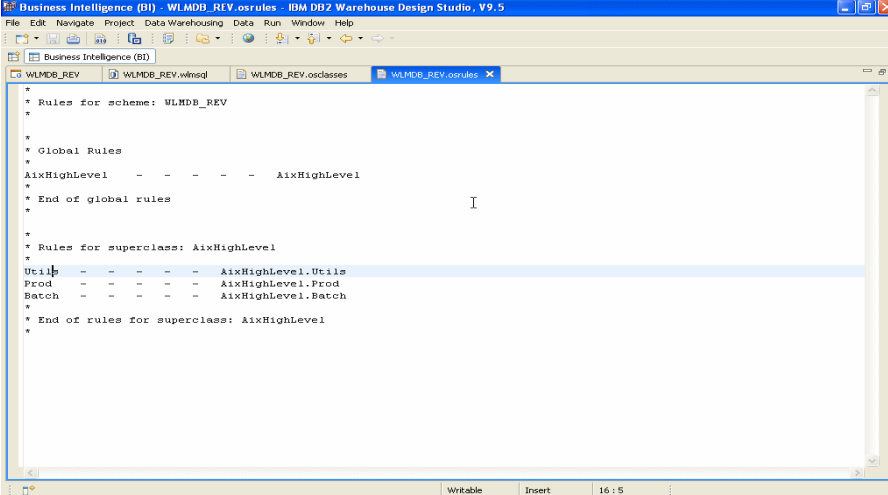
#
# mcklass commands for scheme: WLMDB_REV
#
mcklass -a authuser=root -a adminuser=root -c min=0 -c softmax=100 -c hardmax=100 -c shares=60 aixHighLevel
mcklass -a authuser=root -c shares=20 aixHighLevel.Utils
mcklass -a authuser=root -c shares=60 aixHighLevel.Prod
mcklass -a authuser=root -c shares=20 aixHighLevel.Batch
mcklass -a authuser=root -a adminuser=root aixDefSystem
mcklass -a authuser=root -a adminuser=root aixDefMaint
mcklass -a authuser=root -a adminuser=root aixDefUser
  
```

Figure 10-85 Generated code for Operating system service classes

► **Rules file**

This file is generated and listed when process assignment rules for operating system service classes exist in the scheme. The file is created with scheme name and *.osrules* extension. Example *WLMDB_REV.osrules*

Figure 10-86 on page 317 shows a sample output of Operating systems rules file.



```

Business Intelligence (BI) - WLMDB_REV.osrules - IBM DB2 Warehouse Design Studio, V9.5
File Edit Navigate Project Data Warehousing Data Run Window Help
Business Intelligence (BI)
WLMDB_REV WLMDB_REV.wmsql WLMDB_REV.osclasses WLMDB_REV.osrules x
*
* Rules for scheme: WLMDB_REV
*
*
* Global Rules
*
AixHighLevel - - - - - AixHighLevel
*
* End of global rules
*
*
* Rules for superclass: AixHighLevel
*
Util - - - - - AixHighLevel.Utilm
Prod - - - - - AixHighLevel.Prod
Batch - - - - - AixHighLevel.Batch
*
* End of rules for superclass: AixHighLevel
*

```

Figure 10-86 Generated code sample for Operating system rules file

Note: DB2 WLM artefact code generated by Design Studio is executed by Design Studio against the target database. AIX WLM artefact code generated by Design Studio (the classes script and the rules file) will NOT be processed by Design Studio. Users need to carry the information in these files over to their target AIX machine manually. To implement the code generated for AIX Services classes and rules, you require root authority.

The generated code can be executed from the Design Studio with appropriate connection and authority or it can be shipped to the target source where you like to execute the WLM scheme.



DB2 Workload Manager and DB2 Performance Expert

This chapter provides an introduction to how you can use DB2 Performance Expert in coordination with DB2 Workload Manager functions.

As we described in earlier chapters of this book, DB2 9.5 provides various ways to get information about your workload management (WLM) definitions, and statistics about applications running within the WLM scope. You can write your own reporting scripts to call the table functions, stored procedures, and event monitors. DB2 Performance Expert, on the other hand, performs many of these tasks for you, freeing you to spend your time analyzing the results — not writing and maintaining scripts. In this chapter you can learn where DB2 Performance Expert is the same, similar, or differs from the manual approach.

11.1 DB2 Performance Expert overview

When we think about monitoring a DB2 system, we can consider several areas:

- ▶ Applications
- ▶ Instance and database statistics
- ▶ Configuration
- ▶ Workload management

You can monitor your DB2 system's real time and historical performance behavior using DB2 Performance Expert. DB2 Performance Expert (PE) uses the DB2 snapshot and event monitor capabilities to capture the performance data, and stores the data in DB2 tables on the PE server. You use the DB2 Performance Expert Client to view and work with the data.

Chapter focus

This chapter discusses only the use of DB2 Performance Expert to monitor DB2 Workload Manager capabilities. If you want to learn more about general usage of DB2 Performance Expert, you should consult the following sources:

Author Comment: will these links change for PE V3 ???

- ▶ DB2 Performance Expert product information page at ibm.com:
<http://www-306.ibm.com/software/data/db2imstools/db2tools/db2pe/db2pe-mp.html>
- ▶ DB2 Performance Expert Library page
<http://www-306.ibm.com/software/data/db2imstools/db2tools-library.html#expert>
- ▶ DB2 Performance Expert InfoCenter:
<http://publib.boulder.ibm.com/infocenter/mptoolic/v1r0/topic/com.ibm.db2tools.db2pemp.doc/db2pemphome.htm>
- ▶ IBM Redbooks publication *DB2 Performance Expert for Multiplatforms V2.2*, SG24-6470

How to get DB2 Performance Expert

DB2 Performance Expert is a part of the DB2 V9 Performance Optimization Feature. In DB2 9.1, this included DB2 PE and Query Patroller. In DB2 9.5, the feature includes DB2 PE and the Workload Management Advanced Features

<VERIFY THAT for naming and accuracy. wjc>

DB2 Performance Expert is also available to purchase as a standalone product, and has editions for DB2 Content Manager and DB2 Workgroup Edition. Consult your IBM sales representative for more information, or check out the DB2 Performance Expert home page at

<http://www-306.ibm.com/software/data/db2imstools/db2tools/db2pe/db2pe-mp.html>

Installing and configuring DB2 Performance Expert

We provide the basic steps for setting up DB2 PE in this book. For detail installation and configuration information, refer to *DB2 Performance Expert for Multiplatforms Installation and Configuration Guide*, SC19-1174.

The basic steps are:

1. Install PE Server.
2. Install PE Server FixPack, if applicable.
3. Configure PE Server.
4. Install PE Client.
5. Configure PE Client.

There are no special DB2 Performance Expert setup steps for monitoring DB2 in a multi-partition (DPF) environment, or for monitoring WLM. There are some configuration settings you can modify for frequency of data collection, which is described in “Viewing workload statistics and histograms” on page 343.

For more technical information, hints and tips for WLM monitoring with PE, see “DB2 Performance Expert technical information” on page 350.

11.2 Monitoring your DB2 environment

In this section, we look at a few of the basic monitoring capabilities of DB2 Performance Expert - whether you use WLM or not.

Monitoring applications

When you use DB2 Performance Expert to monitor the applications, or connections, running on your system, you use the Application Summary and Application Details views in the PE Client.

Application Summary

As seen in Figure 11-1, you can view various pieces of information about the running applications in your system. This is called the Application Summary view. The third column shows the Workload ID number. At a glance, then, you are able

to see what activities are running within which workloads. You can also filter, sort, modify, and rearrange which columns appear in the Application Summary page.

CLYDE_DB2INST1 - Application Summary

Application Summary Selected View Tools Window Help

Show Data for: PART0

9/13/07 1:48:47 PM Zoom 0.00:20

9/11/07 10:21:27 AM 9/15/07 5:16:41 PM

DB Name	Application N...	Workload ID	Auth ID	Execution ID	Application Handle	Application Status	User CPU Time...	System CPU Time ...	Host Executio...	Total Sorts	SQL Statement Text
WLMDB	db2bp	7	ADMINSK	adminsk	24,538	UOW/ waiting	0.48:28.865	0.196368	3:17:05.261	7	N/P
WLMDB	dss.exe	7	ADMINMM	adminmm	36,453	UOW/ executing	0.020088	0.003346	0:20:01.406	3	select s_name, s_addr
WLMDB	dss.exe	7	ADMINMM	adminmm	36,559	UOW/ executing	0.006694	0.001466	0:10:36.321	1	select s_name, s_addr
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,644	UOW/ executing	0.155139	0.006394	0:09:46.840	0	select o_orderpriority, si
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,648	UOW/ executing	0.153495	0.006523	0:09:37.004	0	select o_orderpriority, si
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,661	UOW/ executing	0.079632	0.002596	0:09:12.773	0	select * from tpcc.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,663	UOW/ executing	0.003687	0.000985	0:09:08.836	0	select o_orderkey, o_tc
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,676	UOW/ executing	0.114524	0.005453	0:08:56.681	0	select o_orderpriority, si
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,677	UOW/ executing	0.116056	0.005600	0:08:56.637	0	select o_orderpriority, si
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,674	UOW/ executing	0.077657	0.002597	0:08:52.656	0	select * from tpcc.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,669	UOW/ executing	0.080578	0.002501	0:08:52.495	0	select * from tpcc.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,671	UOW/ executing	0.003295	0.000854	0:08:48.722	0	select o_orderkey, o_tc
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,681	UOW/ executing	0.003407	0.000914	0:08:48.523	0	select o_orderkey, o_tc
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,704	UOW/ executing	0.115978	0.005430	0:08:41.996	0	select o_orderpriority, si
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,707	UOW/ executing	0.083850	0.002795	0:08:38.531	0	select * from tpcc.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,702	UOW/ executing	0.079840	0.002413	0:08:38.151	0	select * from tpcc.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,698	UOW/ executing	0.003737	0.000948	0:08:32.766	0	select o_orderkey, o_tc
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,709	UOW/ executing	0.003571	0.000970	0:08:32.433	0	select o_orderkey, o_tc
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,726	UOW/ executing	0.118271	0.005592	0:08:25.463	0	select o_orderpriority, si
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,718	UOW/ executing	0.080485	0.002527	0:08:24.890	0	select * from tpcc.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,715	UOW/ executing	0.079678	0.002396	0:08:23.875	0	select * from tpcc.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,720	UOW/ executing	0.003655	0.000981	0:08:18.651	0	select o_orderkey, o_tc
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,716	UOW/ executing	0.003522	0.000941	0:08:18.049	0	select o_orderkey, o_tc
WLMDB	dss.exe	7	ADMINMM	adminmm	36,486	UOW/ executing	0.013407	0.002813	0:08:15.258	1	select s_name, s_addr
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,732	UOW/ executing	0.117680	0.005746	0:08:10.840	0	select o_orderpriority, si
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,752	UOW/ executing	0.117968	0.005334	0:08:10.619	0	select o_orderpriority, si
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,737	UOW/ executing	0.082364	0.002537	0:08:08.364	0	select * from tpcc.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,744	UOW/ executing	0.078064	0.002426	0:08:06.403	0	select * from tpcc.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,733	UOW/ executing	0.003827	0.000964	0:08:03.794	0	select o_orderkey, o_tc
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,745	UOW/ executing	0.003304	0.000939	0:08:03.749	0	select o_orderkey, o_tc
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,756	UOW/ executing	0.116593	0.005362	0:07:55.154	0	select o_orderpriority, si
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,760	UOW/ executing	0.079382	0.002343	0:07:51.476	0	select * from tpcc.order

Figure 11-1 Application Summary

Application Details

To view details about any one application, double-click the application row in the Application Summary view, and a new window, Application Details, opens, see Figure 11-2. You can look at the currently executing statement, and even launch the DB2 Visual Explain. Some performance counters especially relevant to the workload management functions are the timers and cardinality estimates.

CLYDE_DB2INST1 - Application Details (ADMINMMW*N0.db2inst1.070913183005) on PARTO

Application Details View Tools Window Help

9/13/07 1:48:47 PM Zoom 0:00:20

9/13/07 1:27:42 PM 9/13/07 9:24:41 PM 9/

SQL Statement and Package

Statement

Most recent operation	SQL Fetch
Statement type	Dynamic statement
Section number	15
Application creator	NULLID
Package name	SYSSH100
Package consistency token	SYSLVL01
Cursor name	SQL_CURSH100C15
Blocking cursor	Yes
Node number	0

SQL statement text

```
select s_name, s_address from tpcd.supplier,
tpcd.nation where s_suppkey in ( select
ps_suppkey from tpcd.partsupp where ps_partkey in
( select p_partkey from tpcd.part where p_name
like 'brown%' ) and ps_availqty > ( select 0.5 *
sum(l_quantity) from tpcd.lineitem where
l_partkey = ps_partkey and l_suppkey = ps_suppkey
and l_shipdate >= date ('1997-01-01') and
l_shipdate < date ('1997-01-01') + 1 year ) ) and
s_nationkey = n_nationkey and n_name = 'CHINA'
order by s_name
```

Sort

Statement sorts	0
Total sort time (sec)	0.000000
Sort overflows	0

Rows

Read	25
Written	0
Deleted	0
Updated	0
Inserted	0
Fetches	0

Buffer pool

	Data	Index	XDA
Hit ratio (%)	100.00	100.00	
Logical reads	25	0	
Physical reads	1	0	
Temporary logical reads	0	0	
Temporary physical reads	0	0	

Miscellaneous

SQL compiler cost estimate in timerons	14,008
SQL compiler cardinality estimate	1
Parallelism requested	1
Agents working on statement	1
Subagents created for statement	1

Buttons: Explain View Statement in New Window

Figure 11-2 Application Details - SQL Statement and Package page

The Identification page of Application Details, shown in Figure 11-3, also shows some information that can be useful in troubleshooting your workloads and service classes. The workload ID value is in the top section under the Application Information group heading. The fields named under the heading TP Monitor client are the strings that can be set with the WLM_SET_CLIENT_INFO stored procedure, or on the JDBC™ connection itself. An example of this is described in 4.3.2, “Creating the workloads” on page 68. In this figure, they were not set, so they appear as “N/P” (not present).

Identification

Application information

Handle	36,453
Name	dss.exe
ID	*N0.db2inst1.070913183005
Application status	UDW executing
ID of code page	819
Status change time	9/13/07 1:46:50 PM
Current workload ID	7
Session authentication id	N/P

Database

Input DB alias	CLYDwLM
DB alias	wLMDB
DB name	wLMDB
DB path	/db2data/db2inst1/NODE0000/SQL00001/
Database territory code	1

Client

Operating platform	AIX 64 bit
Communication protocol	Local IPC
Process ID	1,106,184
Product / version ID	SQL09050

TP monitor client

User ID	N/P
Workstation name	N/P
Application name	N/P
Accounting string	N/P

User identification

Authorization ID	ADMINMM
User login ID	adminmm
DRDA correlation token	*N0.db2inst1.070913183005
Coordinating node	0

User authority

	Direct	Indirect
SYSADM	NO	YES
DBADM	NO	NO
CREATETAB	NO	YES
BINDADD	NO	YES
CONNECT	NO	YES
CREATE_NOT_FENC	NO	NO
SYSCTRL	NO	NO
SYSMAINT	NO	NO
IMPLICIT_SCHEMA	NO	YES
LOAD	NO	NO
CREATE_EXT_RT	NO	NO
LIBADM	NO	NO
QUIESCE_CONN	NO	NO
SECADM	NO	NO
SYSQUIESCE	NO	NO
MON	NO	NO

Figure 11-3 Application Details - Identification page

DPF considerations

DB2 Performance Expert provides different views of the work running on your DPF system. If PE detects a multi-partition instance, you will have a dropdown list at the top of the window, where you can select the different views. The views are:

- ▶ Individual partition - shows data for a single partition only.
- ▶ GROUP view - show data from each partition.
Note: This is not related to the DB2 database partition group definitions.
- ▶ GLOBAL view - shows aggregated data from all partitions (uses GLOBAL snapshot).

In Figure 11-4, we see an example of the GROUP view of all applications, sorted by the Total Sorts column, showing the most sorts at the top. The Group view is useful for comparing performance counters between partitions at a glance. You

can display and sort different performance columns quickly see if there are skew.

The screenshot shows the 'Application Summary' window for 'CLYDE_DB2INST1 GROUP'. The window title is 'CLYDE_DB2INST1 - Application Summary'. The menu bar includes 'Application Summary', 'Selected', 'View', 'Tools', 'Window', and 'Help'. Below the menu bar, there are navigation icons and a 'Show Data for:' dropdown set to 'CLYDE_DB2INST1 GROUP'. A timeline at the top shows the current time as 9/13/07 1:28:42 PM, with a zoom level of 0.00:20. The main area contains a table with the following columns: DB Name, Applicatio..., Workload ID, Auth ID, Application Status, User CPU Time..., System CPU Time..., Host Execution Elap..., Total Sorts, SQL Statement Text, Partition, and Coordinating. The table lists various applications like WLMDb, WLMDR, and WLMDB, along with their respective users, statuses, and performance metrics. The SQL Statement Text column shows various queries, including 'select s_name, count(*) as numwait from tpcd.su...' and 'select * from tpcd.orders where o_totalprice<350...'. The Partition and Coordinating columns show 'PART1' through 'PART10'.

DB Name	Applicatio...	Workload ID	Auth ID	Application Status	User CPU Time...	System CPU Time...	Host Execution Elap...	Total Sorts	SQL Statement Text	Partition	Coordinating
WLMDb	dss.exe	7	ADMINMM	UOW waiting	5.061973	0.169640	0.000000	11	select s_name, count(*) as numwait from tpcd.su...	PART1	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	5.083564	0.144410	0.000000	11	select s_name, count(*) as numwait from tpcd.su...	PART2	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	4.733917	0.182125	0.000000	11	select s_name, count(*) as numwait from tpcd.su...	PART3	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	4.775111	0.189107	0.000000	11	select s_name, count(*) as numwait from tpcd.su...	PART4	
WLMDb	db2bp	7	ADMINSK	UOW waiting	0.48,28.865	0.196368	3:17.05.261	7	N/P	PART0	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	3.351249	0.157793	0.000000	5	select supp_nation, cust_nation, l_year, sum(vol...	PART1	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	3.553689	0.225951	0.000000	5	select supp_nation, cust_nation, l_year, sum(vol...	PART2	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	3.413243	0.203533	0.000000	5	select supp_nation, cust_nation, l_year, sum(vol...	PART3	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	3.397663	0.200252	0.000000	5	select supp_nation, cust_nation, l_year, sum(vol...	PART4	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	2.335307	0.327867	0.000000	3	select s_name, s_address from tpcd.supplier, tpc...	PART1	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	5.537929	0.169453	0.000000	3	select nation, o_year, sum(amount) as sum_profi...	PART1	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	2.681061	0.333197	0.000000	3	select s_name, s_address from tpcd.supplier, tpc...	PART2	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	5.905906	0.193709	0.000000	3	select nation, o_year, sum(amount) as sum_profi...	PART2	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	2.364868	0.327496	0.000000	3	select s_name, s_address from tpcd.supplier, tpc...	PART3	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	4.333346	0.216819	0.000000	3	select nation, o_year, sum(amount) as sum_profi...	PART3	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	2.365835	0.320328	0.000000	3	select s_name, s_address from tpcd.supplier, tpc...	PART4	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	4.289165	0.218401	0.000000	3	select nation, o_year, sum(amount) as sum_profi...	PART4	
WLMDb	oltp.exe	11	ADMINMM	UOW waiting	1.016741	0.052164	0.000000	2	select * from tpcd.orders where o_totalprice<350...	PART1	
WLMDb	oltp.exe	7	ADMINMM	UOW waiting	1.753494	0.149520	0.000000	2	select c_count, count(*) as custdist from { select ...	PART1	
WLMDb	oltp.exe	11	ADMINMM	UOW waiting	1.024999	0.055791	0.000000	2	select * from tpcd.orders where o_totalprice<350...	PART2	
WLMDb	oltp.exe	7	ADMINMM	UOW waiting	1.864254	0.114542	0.000000	2	select c_count, count(*) as custdist from { select ...	PART2	
WLMDb	oltp.exe	11	ADMINMM	UOW waiting	0.685570	0.097673	0.000000	2	select * from tpcd.orders where o_totalprice<350...	PART3	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	1.570634	0.044299	0.000000	2	select c_count, count(*) as custdist from { select ...	PART3	
WLMDb	oltp.exe	11	ADMINMM	UOW waiting	0.682922	0.099959	0.000000	2	select * from tpcd.orders where o_totalprice<350...	PART4	
WLMDb	dss.exe	7	ADMINMM	UOW waiting	1.573489	0.039519	0.000000	2	select c_count, count(*) as custdist from { select ...	PART4	
WLMDb	dss.exe	7	ADMINMM	UOW executing	0.012431	0.001544	0:01:03.371	1	select s_name, s_address from tpcd.supplier, tpc...	PART0	
WLMDb	dss.exe	7	ADMINMM	UOW executing	0.071761	0.002569	0:01:44.863	1	select s_name, count(*) as numwait from tpcd.su...	PART0	
WLMDb	db2stmm	0	ADMINTI	Appl. has been d...	0.037635	0.003273	0.000000	0	N/P	PART0	
WLMDb	oltp.exe	11	ADMINMM	UOW executing	0.075044	0.001825	46.806501	0	select * from tpcd.orders where o_totalprice<350...	PART0	
WLMDb	db2wlm	0	ADMINSK	connect completed	2.543053	5.393999	0.000000	0	N/P	PART0	
WLMDb	dss.exe	7	ADMINMM	UOW executing	0.005077	0.000661	0:01:43.130	0	select nation, o_year, sum(amount) as sum_profi...	PART0	
WLMDR	rh2taskd	0	ADMINSK	connect completed	0.001956	0.003693	0.000000	0	N/P	PART0	

Figure 11-4 Application Summary - DPF - Group view

When you drill down to the Application Details in a DPF application, you can use the Subsections page to view how a query is progressing. Figure 11-5 shows an example.

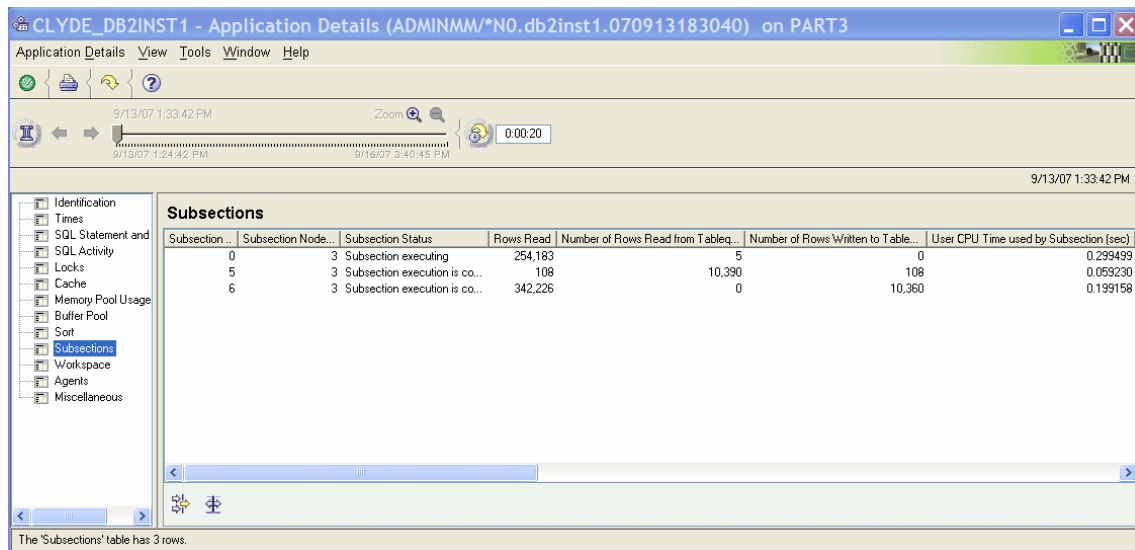


Figure 11-5 Application Details - DPF - Subsections

11.2.1 Monitoring instance and database statistics

You can use DB2 Performance Expert to monitor the overall instance and system performance by looking at database objects such as buffer pools, table spaces, tables, and so on. We do not show all the features here. We only show some typical or important examples that could be relevant to a WLM environment.

Heavy-hitter tables

You can use the Statistics Details - Tables view to see which tables are being accessed the most. In Figure 11-6, we see the data for only Partition 1, sorted by Rows Read, and we can see the LINEITEM table is by far the most-read table.

CLYDE_DB2INST1 - Statistics Details

Statistics Details View Tools Window Help

Show Data for: PART1

9/13/07 2:27:32 PM Zoom 0.00:20

9/11/07 10:25:27 AM 9/16/07 3:54:45 PM 9/13/07 2:27:32 PM

DB Name	Table Scheme...	Table Name	Table Space ID	Rows Written	Rows Read	Overflowed Rows	Table Type	Page Reorg	Data Object Pages	Inc
WLMDB	TPCD	LINEITEM	7	0	1,012,168,001	0	User table	0	40,432	
WLMDB	TPCD	ORDERS	9	0	551,712,501	0	User table	0	38,512	
WLMDB	TPCD	PART	11	0	238,240,906	0	User table	0	3,065	
WLMDB	TPCD	CUSTOMER	11	0	17,123,547	0	User table	0	418	
WLMDB	TPCD	PARTSUPP	11	0	11,282,498	0	User table	0	11,488	
WLMDB	TPCD	SUPPLIER	11	0	1,592,058	0	User table	0	26	
WLMDB	ADMINHM	SCSTATS_BASIC_MON	16	0	1,111,836	0	User table	0	96	
WLMDB	PESRVUSR	HISTOGRAMBIN_CETUS__PEINST	2	291,760	210,211	0	User table	750	32	
WLMDB	ADMINHM	WLSTATS_BASIC_MON	16	0	12,266	0	User table	0	37	
WLMDB	PESRVUSR	SCSTATS_CETUS__PEINST	2	2,136	2,136	0	User table	12	1	
WLMDB	PESRVUSR	WLSTATS_CETUS__PEINST	2	1,068	1,068	0	User table	2	1	
WLMDB	ADMINHM	ACTIVITYSTMT_ACT_MON	2	0	360	0	User table	0	3	
WLMDB	PESRVUSR	WCSTATS_CETUS__PEINST	2	356	356	0	User table	0	1	
WLMDB	ADMINHM	ACTIVITY_ACT_MON	2	4,997	0	0	User table	0	238	
WLMDB	ADMINHM	CONTROL_ACT_MON	2	4	0	0	User table	0	2	
WLMDB	PESRVUSR	CONTROL_CETUS__PEINST	2	2	0	0	User table	0	1	
WLMDB	ADMINHM	CONTROL_THRESH_MON	2	2	0	0	User table	0	1	

The 'Tables' table has 17 rows.

Figure 11-6 Statistics Details - Heavy-hitter tables for Partition 1

Most costly statements

In the Statistics Details - Dynamic SQL view, you can see the statements in the statement cache. By sorting on different columns, you can quickly see which statements are the most costly in CPU time, the most executed, the longest-running and so on. In Figure 11-7, we see the Dynamic SQL page for Partition 1, sorted by the Average time per execution, which shows us the longest-running statements at the top.

CLYDE_DB2INST1 - Statistics Details

Statistics Details View Tools Window Help

Show Data for: PART1

9/13/07 2:27:32 PM Zoom 0.00:20

9/12/07 7:34:00 PM 9/14/07 7:34:35 AM 9/13/07 2:27:32 PM

Dynamic SQL Statements

DB Name	Statement	Executions	Elapsed exec. tim.	Avg. time per exe.	Sorts	Total user CPU time (sec)	Total system CPU time (s.)	A
WLMDB	select s_name, s_address from tpcd.supplier, tpcd.nation wher...	24	5.44:29.836	0:14:21.243	46	0:01:07.932	8.467310	
WLMDB	select sum(L_extendedprice) / 7.0 as avg_yearly from tpcd.linei...	15	3:18:16.873	0:13:13.125	15	28.606925	1.049903	
WLMDB	select L_shipmode, sum(case when o_orderpriority = '1-URGE...	20	4:13:10.000	0:12:39.550	15	33.482433	2.415236	
WLMDB	select o_year, sum(case when nation = 'ROMANIA' then volu...	57	11:35:53.288	0:12:12.514	96	0:01:47.593	7.328404	
WLMDB	select s_name, count(*) as numwait from tpcd.supplier, tpcd.lin...	20	3:58:09.042	0:11:54.452	30	0:01:20.427	2.420144	
WLMDB	select s_name, count(*) as numwait from tpcd.supplier, tpcd.lin...	57	11:11:10.116	0:11:46.493	96	0:04:11.721	7.808597	
WLMDB	select c_name, c_custkey, o_orderkey, o_orderdate, o_totalpr...	56	10:48:45.593	0:11:35.100	47	0:01:44.553	4.534597	
WLMDB	select s_name, s_address from tpcd.supplier, tpcd.nation wher...	24	4:25:11.613	0:11:02.984	48	34.303732	5.489589	
WLMDB	select nation, o_year, sum(amount) as sum_profit from (select ...	20	3:33:41.378	0:10:41.069	30	37.867546	1.903107	
WLMDB	select s_name, s_address from tpcd.supplier, tpcd.nation wher...	66	11:00:53.216	0:10:00.806	66	0:02:59.918	17.725987	
WLMDB	select L_orderkey, sum(L_extendedprice * (1 - L_discount)) as re...	15	2:29:18.764	0:09:57.251	28	31.319942	2.041769	
WLMDB	select s_name, s_address from tpcd.supplier, tpcd.nation wher...	17	2:35:21.427	0:09:08.319	30	37.557366	5.716454	
WLMDB	select c_name, c_custkey, o_orderkey, o_orderdate, o_totalpr...	24	3:28:13.098	0:08:40.546	16	35.425351	1.165458	
WLMDB	select s_name, s_address from tpcd.supplier, tpcd.nation wher...	66	8:48:54.606	0:08:00.827	124	0:02:18.272	14.684260	
WLMDB	select s_name, s_address from tpcd.supplier, tpcd.nation wher...	61	8:04:18.035	0:07:56.361	94	0:02:18.726	14.858869	
WLMDB	select c_custkey, c_name, sum(L_extendedprice * (1 - L_disco...	16	2:03:36.484	0:07:43.530	28	25.139854	0.806514	
WLMDB	select nation, o_year, sum(amount) as sum_profit from (select ...	48	5:43:45.717	0:07:09.702	94	0:01:53.695	4.956105	
WLMDB	select c_count, count(*) as custdist from (select c_custkey, c...	24	2:51:16.064	0:07:08.169	48	20.159285	1.358429	
WLMDB	select L_shipmode, sum(case when o_orderpriority = '1-URGE...	47	5:32:11.384	0:07:04.072	47	0:01:36.628	6.142481	
WLMDB	select o_year, sum(case when nation = 'IRAQ' then volume el...	24	2:42:03.251	0:06:45.135	40	43.028362	1.460173	
WLMDB	select o_orderpriority, count(*) as order_count from tpcd.orders...	51	5:44:50.706	0:06:45.700	47	0:01:17.376	3.225784	
WLMDB	select s_name, s_address from tpcd.supplier, tpcd.nation wher...	66	7:18:53.793	0:06:38.997	132	0:02:25.216	15.583924	
WLMDB	select L_orderkey, sum(L_extendedprice * (1 - L_discount)) as re...	66	6:59:10.229	0:06:21.064	102	0:01:53.807	9.303443	
WLMDB	select L_orderkey, L_linenumbr from tpcd.lineitem where L_part...	9	0:56:41.793	0:06:17.977	0	14.489193	0.788810	
WLMDB	select c_custkey, c_name, sum(L_extendedprice * (1 - L_disco...	65	6:29:55.942	0:05:59.938	112	0:01:39.883	4.236322	
WLMDB	select s_name, s_address from tpcd.supplier, tpcd.nation wher...	15	1:28:14.189	0:05:52.946	28	32.247769	3.861133	
WLMDB	select s_name, s_address from tpcd.supplier, tpcd.nation wher...	47	4:11:59.112	0:05:21.683	94	0:02:00.566	11.178312	
WLMDB	select sum(L_extendedprice) / 7.0 as avg_yearly from tpcd.linei...	47	4:11:06.637	0:05:20.567	47	0:01:24.713	2.616301	
WLMDB	select o_orderpriority, count(*) as order_count from tpcd.orders...	24	2:03:45.141	0:05:09.381	20	33.896968	1.189631	
WLMDB	select n_name, sum(L_extendedprice * (1 - L_discount)) as reve...	24	2:03:47.720	0:05:09.488	20	36.245449	1.142164	

Receive statement cache information

The Dynamic SQL Statements' table has 154 rows.

Figure 11-7 Dynamic SQL - sorted by Average time per execution - Partition 1

Buffer pool hit ratio

In Figure 11-8, we see the buffer pool hit ratios for all the defined buffer pools, across all the partitions. Using the Group view in this case is a quick way to see the partitions at a glance.

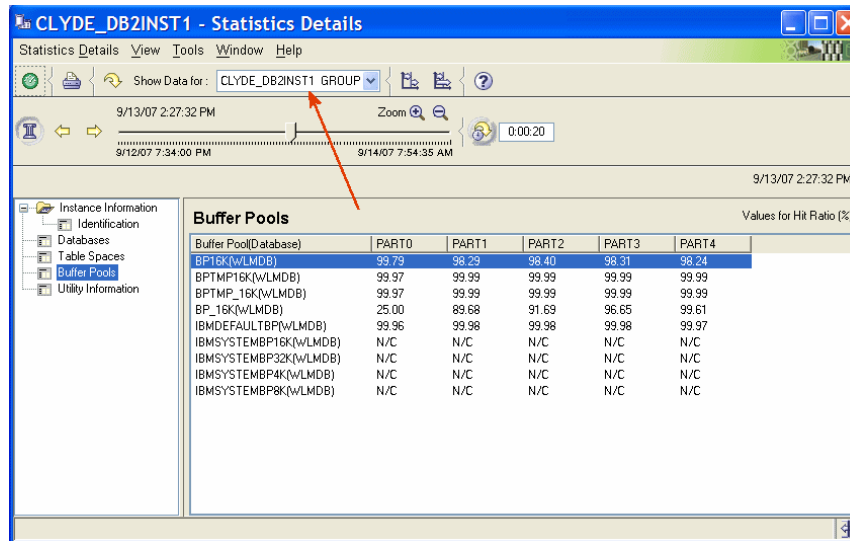


Figure 11-8 Statistics Details - Buffer pool hit ratio - Group view

11.3 Monitoring DB2 Workload Manager

DB2 Performance Expert introduces new monitoring capabilities to coincide with the DB2 9.5 workload management features. We describe those capabilities in this section.

PE V3.1 uses only the Statistics, not the Activities, event monitor to capture WLM performance data. The Statistics event monitor captures statistics that are measured over a set period of time. Compared to Statement or Activities event monitors, the Statistics event monitor is an inexpensive method of capturing historical information because this type of event monitor deals with aggregated activity information instead of individual activities and you can target it to a single service class or work class. Within PE, the Statistics event monitor data is written to tables in the monitored database, and PE retrieves the data into PE performance database, and removes it from the monitored database. You do not have to keep track of the event monitor table growth because PE keeps it cleared out.

11.3.1 Workload Management Key Performance Indicators

The DB2 Performance Expert System Overview shows Key Performance Indicators (KPIs) for many common performance counters. Counters are

grouped and shown in “perflets” on the System Overview. One of the perflets is for Workload Management. In Figure 11-9, we see the most recent statistics captured for WLM statistics for partition 0. The counters are sorted by the “worst” at the top, which may vary by the type of counter.

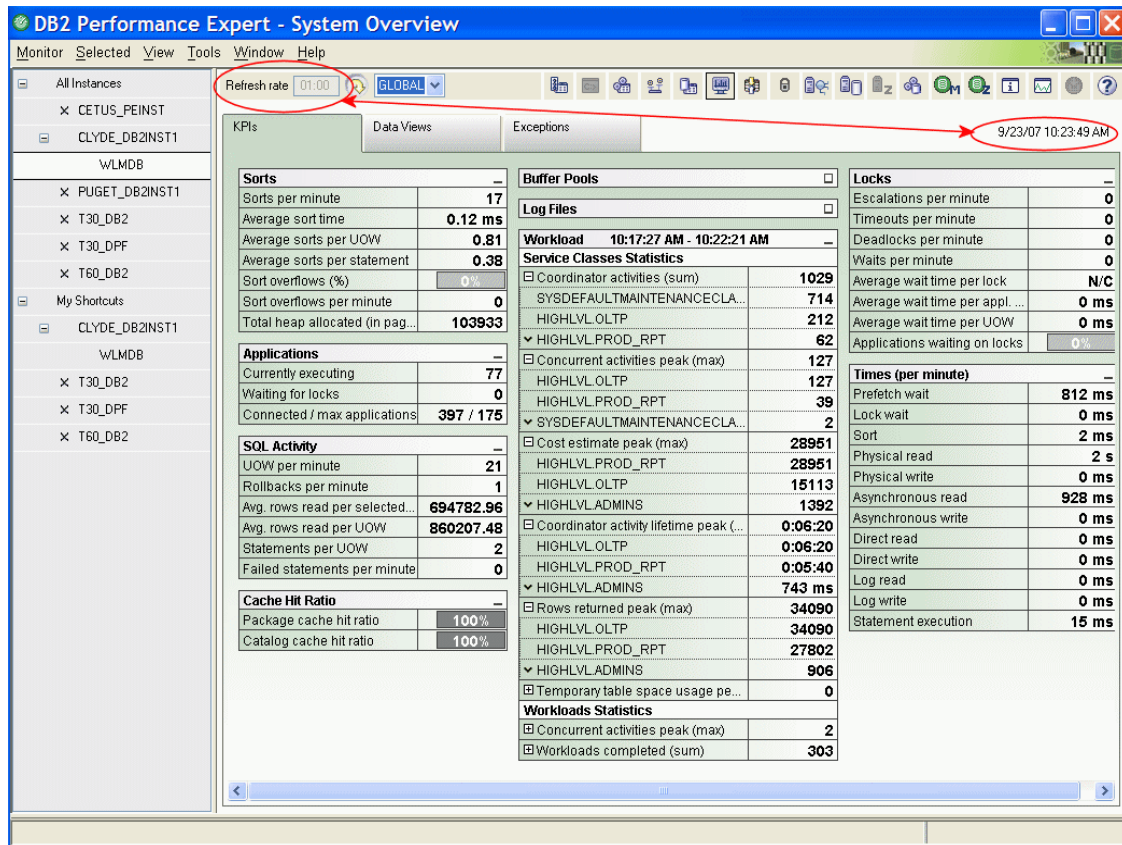


Figure 11-9 System Overview - Workload Manager perflet

Most PE performance counters are collected from DB2 snapshots. The WLM statistics, however, are collected only via the statistics event monitor and are collected at a different interval than the snapshot counters. This is why you see the time interval shown on the top of the WLM perflet - to let you know the interval over which the statistics data were collected and that it may not match the time stamp shown at the top-right of the window which is controlled by a different refresh rate.

In Figure 11-9, we can see the System Overview refresh rate has been set to 1 minute, and this is what controls the time stamp at the top-right of the window.

The WLM collection interval is, however, specified elsewhere and we can tell it was set to 5 minutes (10:17:27 AM - 10:22:21 AM).

We discuss how to configure the WLM collection interval in “Monitoring non-default workloads” on page 341.

By watching the Workload KPIs, you can always have a current view of which workloads are active and busy. If you need to investigate more about the workloads, you can look at the Workload Management screens in PE.

11.3.2 Viewing workload management definitions

To view Workload Management detail data in PE, you must click on the Workload Management icon on the Toolbar on the System Overview page, as shown in Figure 11-10.



Figure 11-10 DB2 Performance Expert Toolbar

The first screen that appears lists each monitored database, with counts of the various WLM objects defined in the database. An example is shown in Figure 11-11. In our lab setup, we are only monitoring one database in the instance. To drill down to more details for the WLMDB, we double-click on the WLMDB.

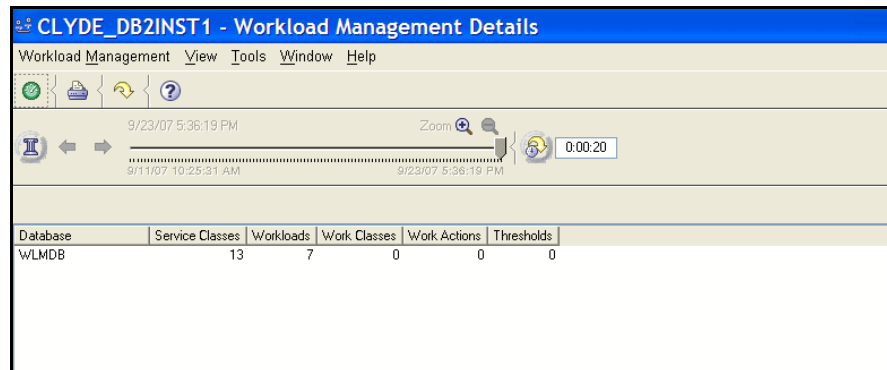


Figure 11-11 Workload Management Details

Let's look at the same WLM definitions as described for the mixed workload in 7.3, "Manage the work" on page 176. Here we have one top-level service class with several subclasses and workloads underneath.

The definitions for the mixed workload are shown in Figure 11-12. When you select the Service Classes in the upper part of the screen, its associated subclasses are highlighted in the lower part. In our mixed workload, we have several subclasses all belonging to the HIGHLVL service class.

The columns displayed in the definition view can be rearranged or sorted. In our case we have arranged the subclasses to show the common attributes such as the values that were specified on the COLLECT AGGREGATE clause. It is an easy way to see all our definitions at a glance.

The screenshot shows the 'Service Classes' section of the Workload Management interface. It contains two tables: 'Service Superclasses: Definitions' and 'Service Subclasses: Definitions'.

Service Superclasses: Definitions

Service Superclass	Status	Agent Priority	Prefetch Priority	Outbound Correlator	Created	Last Updated	Identifier	Comment	Database
SYSDEFAULTSYSTEMCLASS	Enabled	-32,768	Default	N/P	8/14/07 5:09:16 PM	9/21/07 4:19:09 PM	1	N/P	wLMDB
SYSDEFAULTMAINTENANCECLASS	Enabled	-32,768	Default	N/P	8/14/07 5:09:16 PM	9/21/07 4:19:09 PM	2	N/P	wLMDB
SYSDEFAULTUSERCLASS	Enabled	-32,768	Default	N/P	8/14/07 5:09:16 PM	9/21/07 4:19:08 PM	3	N/P	wLMDB
HIGHLVL	Enabled	-32,768	Default	N/P	9/22/07 5:27:06 PM	9/22/07 5:27:11 PM	14	N/P	wLMDB

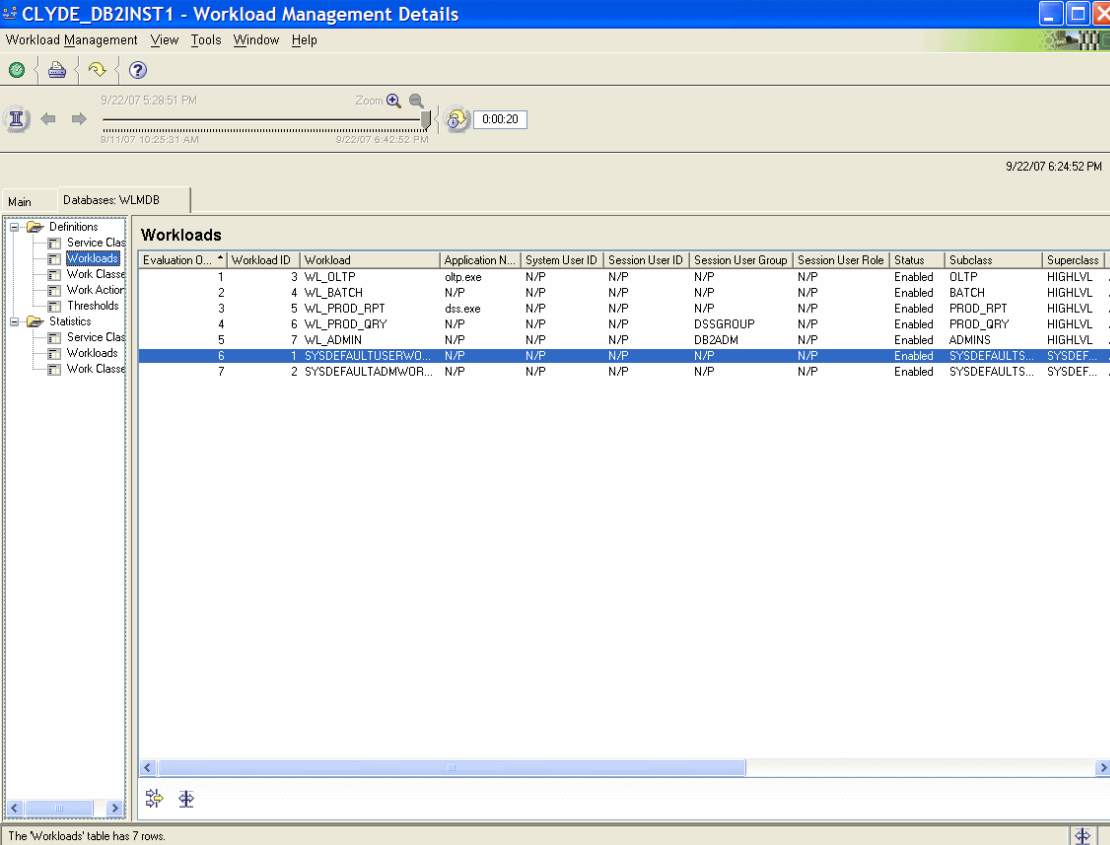
Service Subclasses: Definitions

Service Subclass	Status	Agent Priority	Prefetch Pri...	Last Updated	Collect Activity Data on	Collect Aggregate Activity D...	Collect Aggregate Request...
SYSDEFAULTSYSTEMCLASS.SYSD...	Enabled	-32,768	Default	8/14/07 5:09:16 PM	Coordinator partition	None	None
SYSDEFAULTMAINTENANCECLASS...	Enabled	-32,768	Default	8/14/07 5:09:16 PM	Coordinator partition	None	None
SYSDEFAULTUSERCLASS.SYSEF...	Enabled	-32,768	Default	8/14/07 5:09:16 PM	Coordinator partition	None	None
HIGHLVL.SYSDEFAULTSUBCLASS	Enabled	-32,768	Default	9/22/07 5:27:06 PM	Coordinator partition	None	None
HIGHLVL.ADMINS	Enabled	-32,768	Default	9/22/07 5:27:12 PM	Coordinator partition	Base	None
HIGHLVL.BATCH	Enabled	-32,768	Default	9/22/07 5:27:12 PM	Coordinator partition	None	Base
HIGHLVL.PROD_RPT	Enabled	-32,768	Default	9/22/07 5:27:12 PM	Coordinator partition	Extended	Base
HIGHLVL.PROD_QRY	Enabled	-5	Default	9/22/07 5:27:12 PM	Coordinator partition	Extended	Base
HIGHLVL.QLTP	Enabled	-10	Default	9/22/07 5:27:13 PM	Coordinator partition	Extended	Base

The 'Service Classes' table has 4 rows.

Figure 11-12 WLM Service class definitions - mixed workload

Next we want to view the definitions for the workloads, so we select Workloads from the navigation tree on the left side of the window. In Figure 11-13, we can see all the workloads for the WLMDB database, sorted by the evaluation order. In Chapter 7, “WLM sample scenarios - Mixed OLTP and DSS environment” on page 173, the OLTP workload was added and placed ahead of the other workloads in the evaluation order, and indeed that is what we see here.



Evaluation O...	Workload ID	Workload	Application N...	System User ID	Session User ID	Session User Group	Session User Role	Status	Subclass	Superclass
1	3	WL_OLTP	oltp.exe	N/P	N/P	N/P	N/P	Enabled	DLTP	HIGHLVL
2	4	WL_BATCH	N/P	N/P	N/P	N/P	N/P	Enabled	BATCH	HIGHLVL
3	5	WL_PROD_RPT	dss.exe	N/P	N/P	N/P	N/P	Enabled	PROD_RPT	HIGHLVL
4	6	WL_PROD_QRY	N/P	N/P	N/P	DSSGROUP	N/P	Enabled	PROD_QRY	HIGHLVL
5	7	WL_ADMIN	N/P	N/P	N/P	DB2ADM	N/P	Enabled	ADMINS	HIGHLVL
6	1	SYSDEFAULTUSERW...	N/P	N/P	N/P	N/P	N/P	Enabled	SYSDEFAULTS...	SYSDEF...
7	2	SYSDEFAULTADMWOR...	N/P	N/P	N/P	N/P	N/P	Enabled	SYSDEFAULTS...	SYSDEF...

Figure 11-13 WLM Workload definitions - mixed workload

To view detail about any WLM definition, just double-click it. Figure 11-14 shows an example detail page for the WL_PROD_QRY workload.

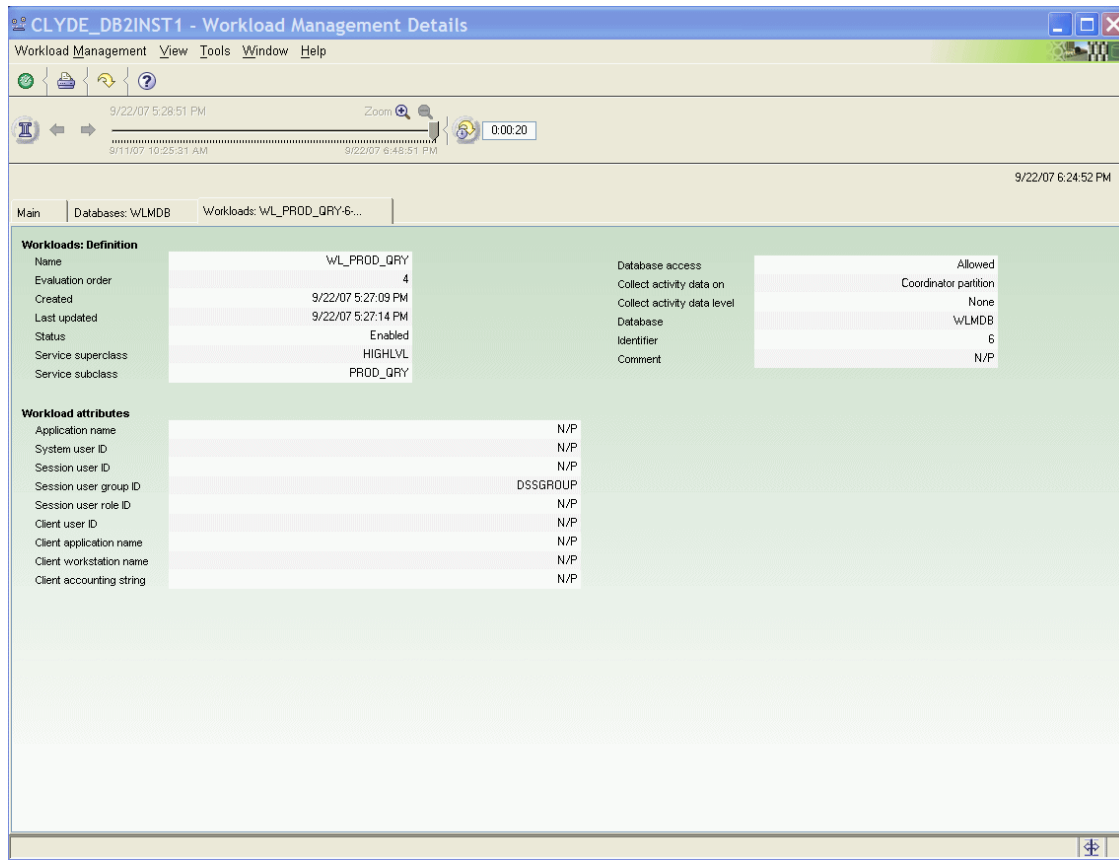


Figure 11-14 Workload definition details

We can use Performance Expert history mode to see what the definitions were in the past.

11.3.3 Viewing Workload Management statistics

In Chapter 4, Chapter 5, and Chapter 7, we saw examples of how to capture statistical data for workloads. In this section, we look at how PE can do this.

Using PE to monitor the default WLM environment

If you do not configure WLM service classes, workloads, and so on, but you do use DB2 Performance Expert, you will still be able to view the base statistics counts that are available. The manual method is described in 3.5.2, “Monitoring the default WLM environment” on page 55, where you can write a query to get

information. In PE, you can open the WLM Statistics page. In Figure 11-15, we see that PE can display the same information you got from the query in Example 3-6 on page 56, to view the high watermark, or peak, connections within the default service classes.

The same screen also displays the service subclasses. When you select the superclass, the associated subclass(es) will be highlighted.

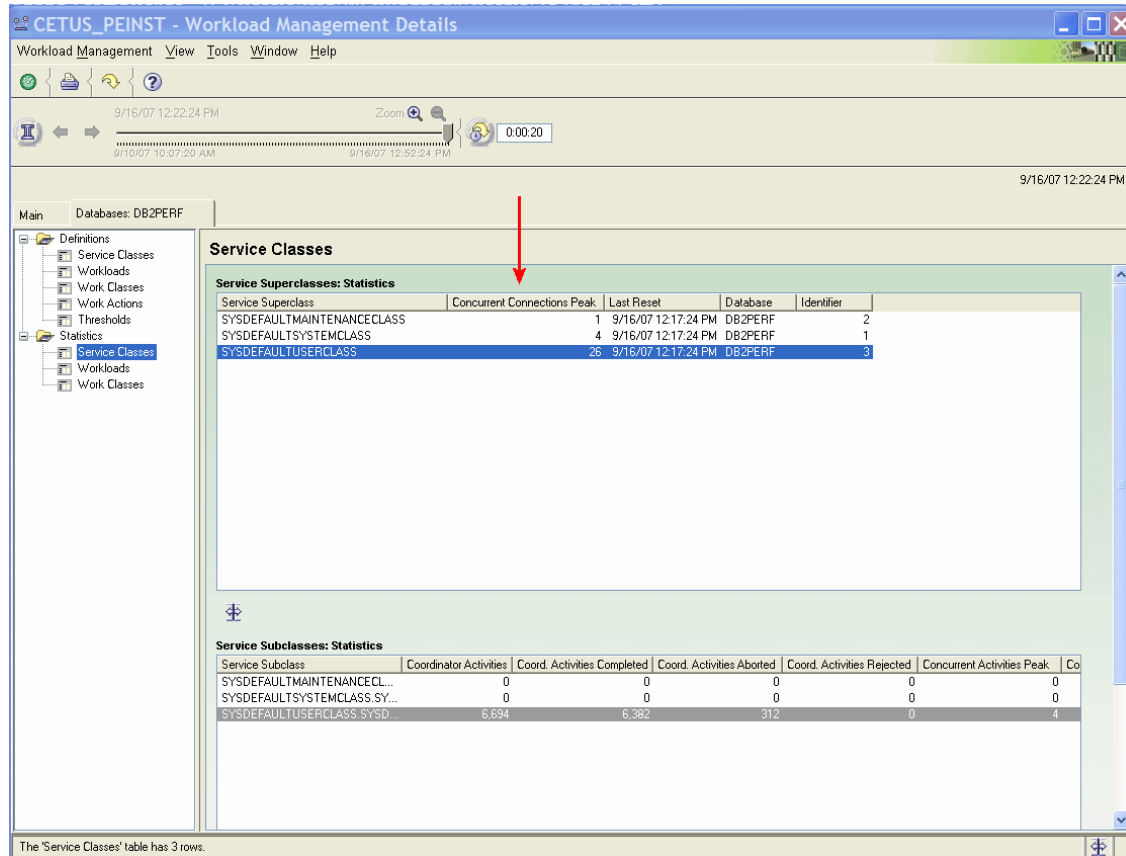


Figure 11-15 WLM default Service Class statistics

To see more detail about the subclass statistics, double-click its entry on the table in the lower portion of the screen. This launches a new tab, as shown in Figure 11-16. Here we see the same information as on the previous screen but for the single subclass. Since we have not enabled any of the collection parameters on the service classes, many fields are reported as -1, meaning the data is not present.

Notice also that we also have a section named Histograms. In this case there is no histogram data because no collection has been activated yet. We see more about histograms in “Viewing workload statistics and histograms” on page 343.

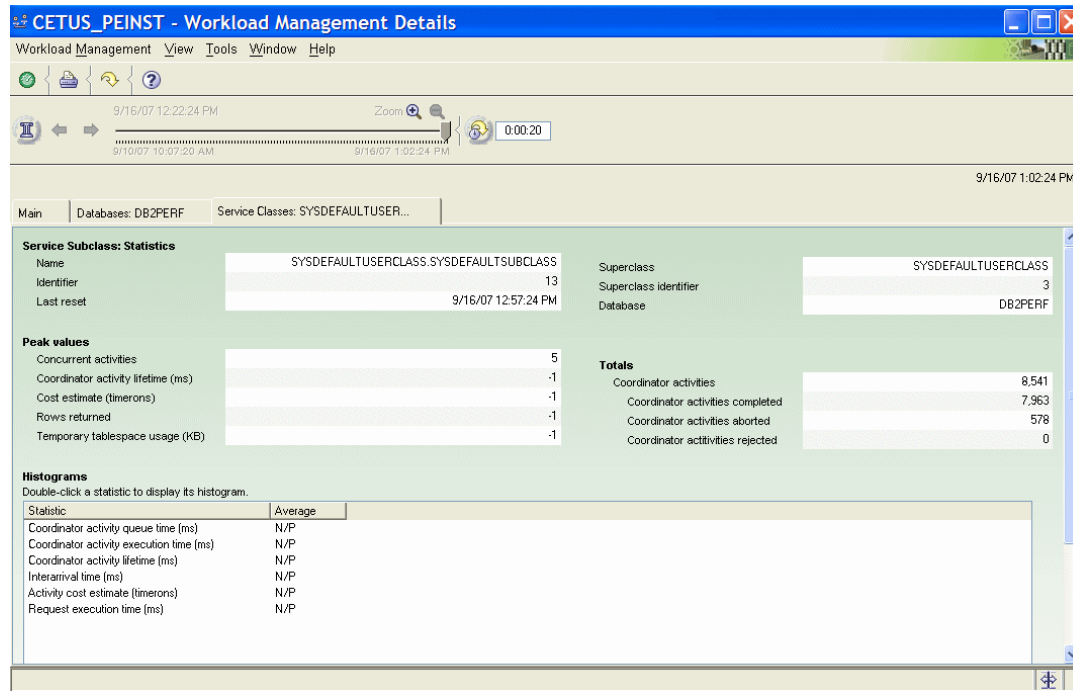


Figure 11-16 WLM default Subclass statistics details

You can view default workload statistics by selecting Workload from the navigation tree on the Workload Management Details page, as shown in Figure 11-17.

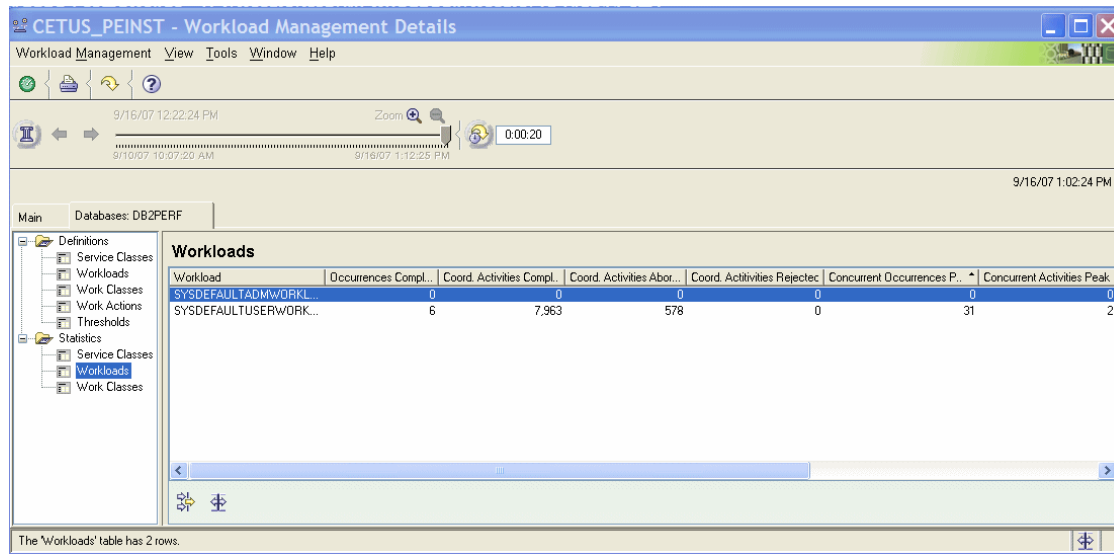


Figure 11-17 WLM default Workload statistics

DPF Mode

When you are using a DPF system, you have some other options for viewing the statistics. We look at a default DPF system where, as in the previous examples, we have not configured any WLM settings. In Figure 11-18, we see the same type of information as we saw in Figure 11-15 on page 335, but instead this is showing counts only for the designated partition - PART0.

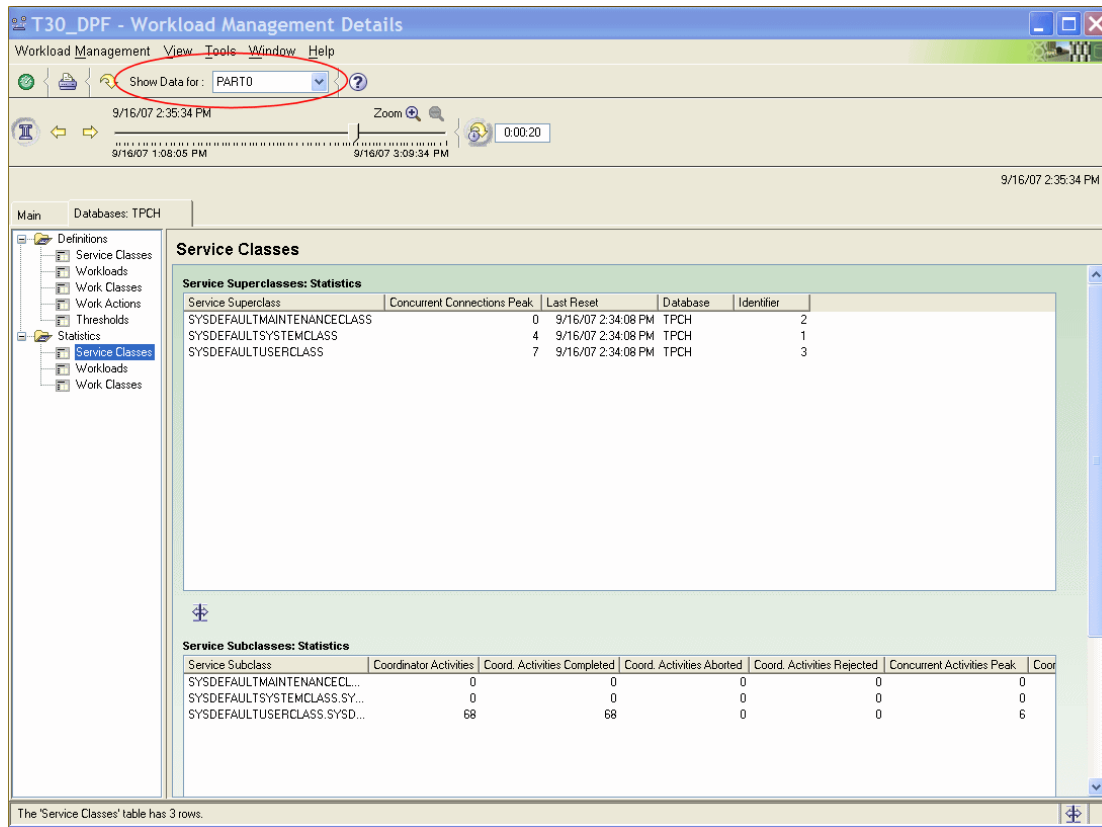


Figure 11-18 WLM default Service Class statistics - DPF- Partition 0

You can choose other views from the dropdown list. Let's look at the GLOBAL view next, in Figure 11-19. We selected GLOBAL from the list and in this case the counts did not change.

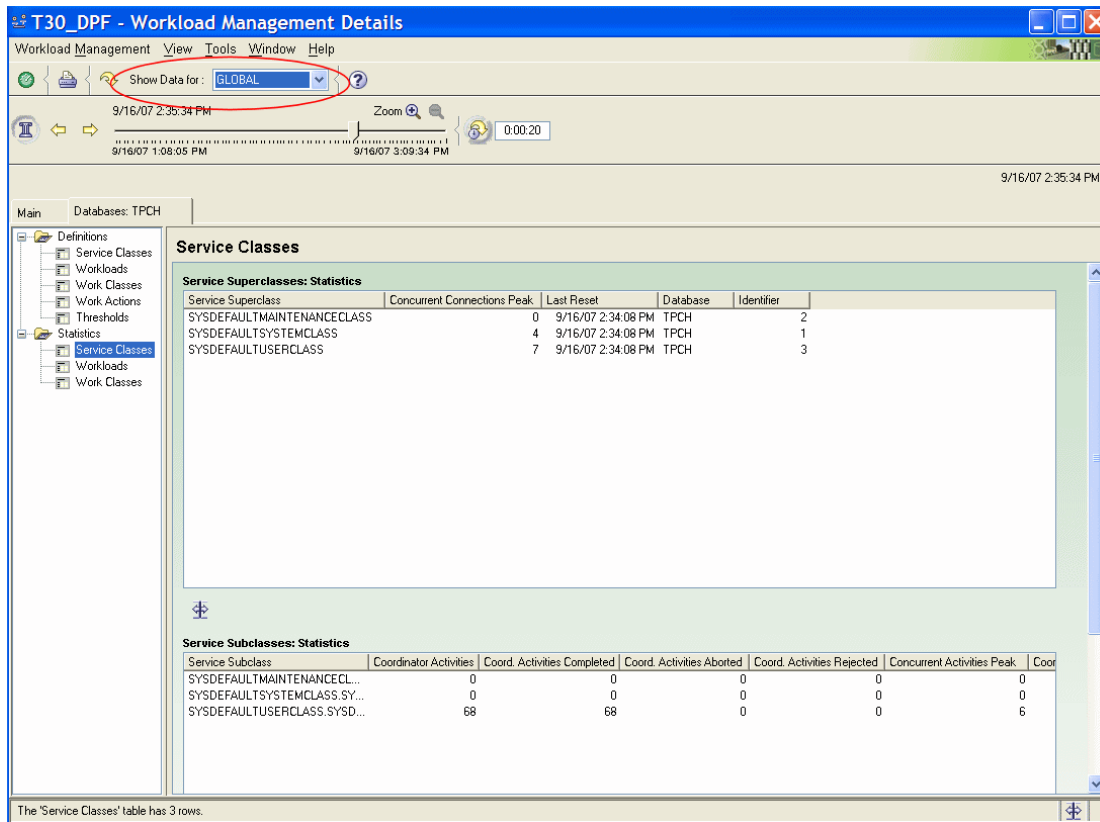


Figure 11-19 WLM default Service Class statistics - DPF- GLOBAL view

In Figure 11-20, we see the results of choosing the GROUP option from the dropdown list box. In this view, we can see the key counts for each partition. You can't see all the counts for all partitions on this page, so the most critical ones are shown here.

Note: This view may not be available on systems with a very large number of partitions.

Note to Reviewer: Question for PE dev: is that stmt correct about not being able to view wht many partitions? I think I saw this in other parts of PE at some customer site. What is the max?

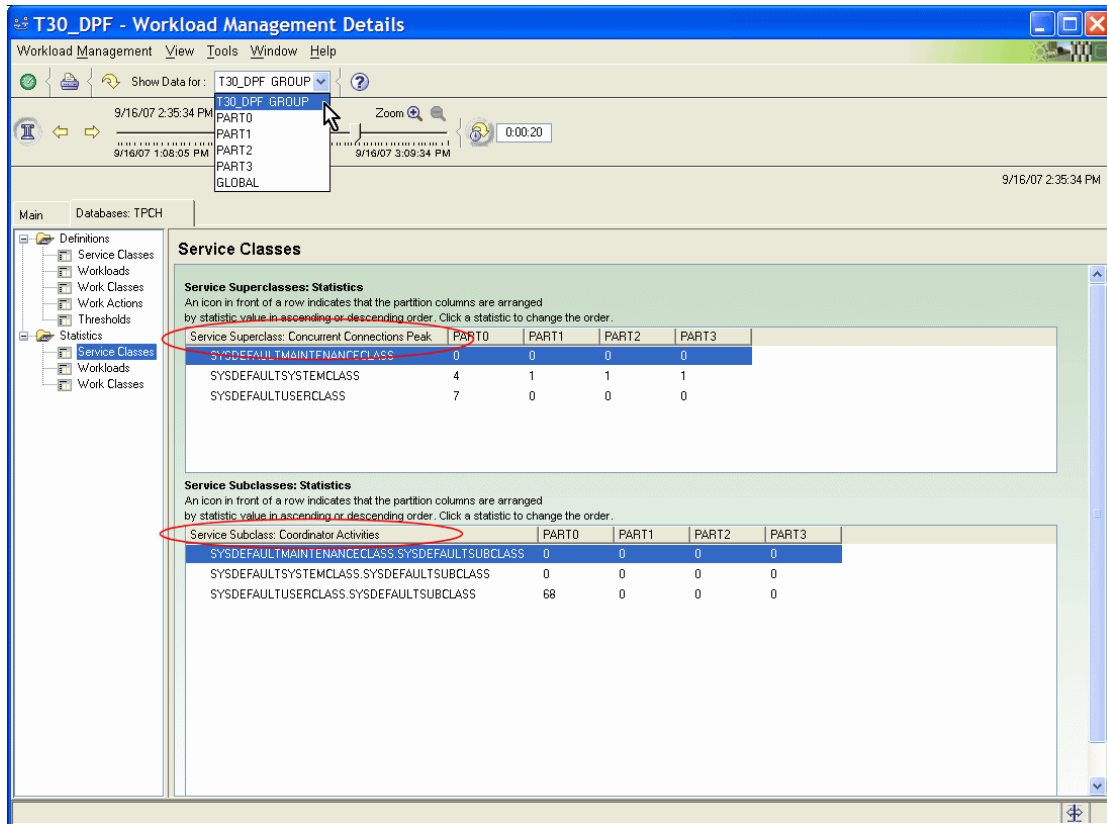


Figure 11-20 WLM default Service Class statistics - DPF- GROUP view

Double-click a Service Class to view the counts for all partitions for that one service class, as shown in Figure 11-21.

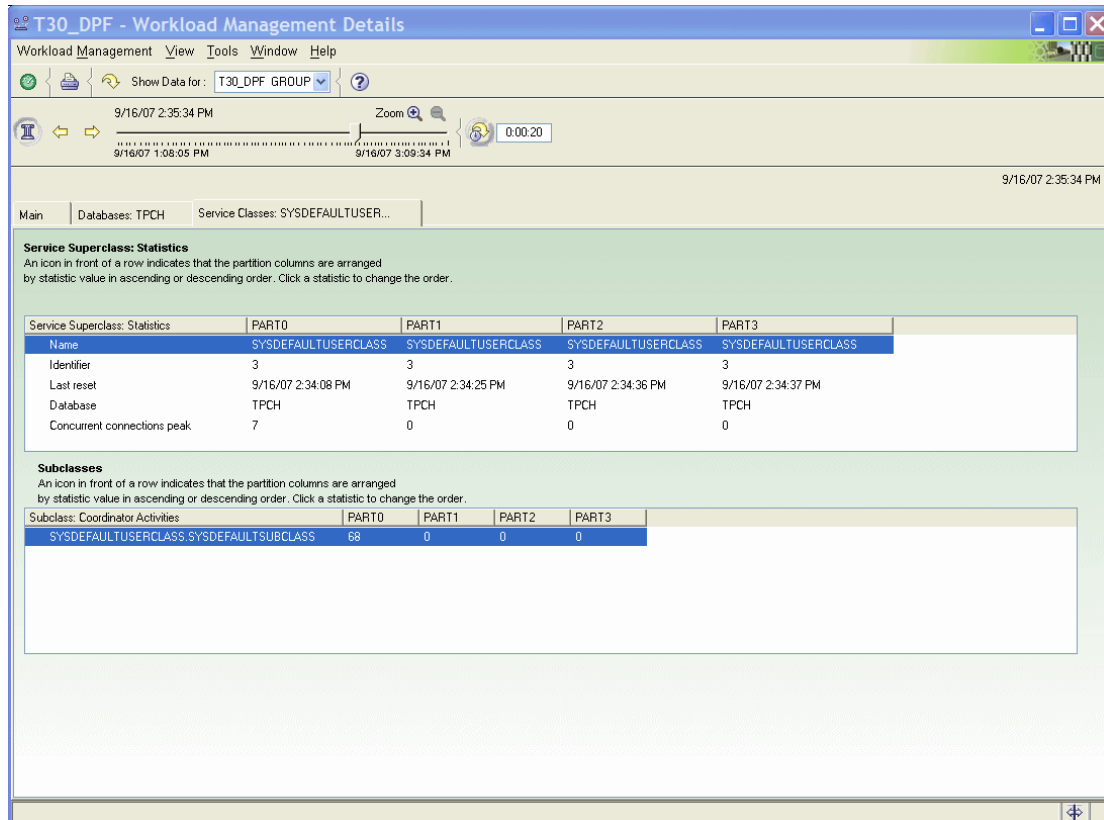


Figure 11-21 WLM default Service Class statistics - DPF- GROUP details view

Monitoring non-default workloads

In earlier chapters of this book, we saw how to write queries against the statistics event monitor tables to get information about the workload performance. Let's see how we can do that with DB2 Performance Expert.

When you monitor the statistics with DB2 PE, you do not need to create and maintain the statistics event monitor yourself - PE does this. You also do not need to write all your own queries to get at the information.

Let us assume we have the similar workload that has run against our WLMDDB database in 7.4.2, "Monitoring and analyzing the service classes" on page 185, or in Chapter 5, "Monitoring" on page 83. We are using DB2 Performance Expert to set up and collect the data from the statistics event monitor. The WLM_COLLECT_INT database configuration value has been set to 0, which

allows PE to control its own data collection. We did *not* create the BASIC_MON event monitor as described in the earlier chapters.

In the PE monitored instance properties, we define the collection interval to be 5 minutes for both workload definitions and workload statistics, as shown in Figure 11-22.

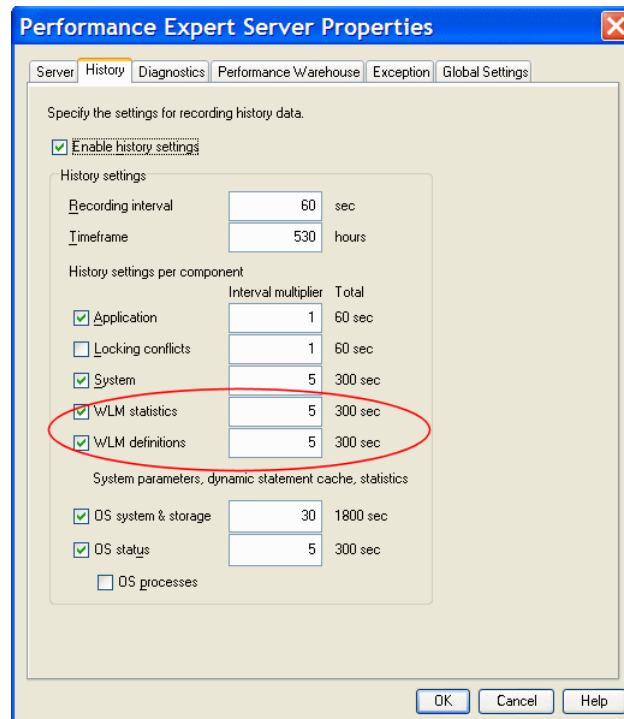


Figure 11-22 Setting the WLM statistics collection interval in DB2 Performance Expert

That is all the setup that is required. PE will create a statistics event monitor in the monitored database and it will handle the data retrieval. You can read more about the technical details in “DB2 Performance Expert technical information” on page 350.

Following are some examples of screens where you can quickly see the WLM statistics without writing the queries as described in earlier chapters of this book.

Observe running applications by workload ID

In Figure 11-23, we see the Application Summary showing all the database connections, sorted by the Workload ID. We can see most of them are in either workload 3 or 5. We know from looking at the WLM definitions earlier

Figure 11-13 on page 333, that workload 3 is WL_OLTP, and workload 5 is the WL_PROD_RPT workload.

PID	DB Name	Application Name	Workload	Auth ID	Application Status	User CPU Time...	System CPU Time ...	Host Execution Elap...	Seq#	Total Sorts	Total Sort Time (sec)	SQL Statement Text
655,436	WLMD8	oltp.exe	3	ADMINMM	UO'w executing	3,971,258	0.167764	0:02:45.340	00007	10	0.001348	select o_orderprio
917,818	WLMD8	oltp.exe	3	ADMINMM	UO'w executing	5,285,468	0.184209	0:02:39.194	00008	14	0.000447	select o_orderprio
2,314,474	WLMD8	oltp.exe	3	ADMINMM	UO'w executing	3,946,053	0.163578	0:02:30.020	00007	10	0.001572	select o_orderprio
1,392,994	WLMD8	oltp.exe	3	ADMINMM	UO'w executing	3,857,434	0.150986	0:02:26.683	00007	8	0.000848	select * from lpcd.o
2,023,874	WLMD8	oltp.exe	3	ADMINMM	UO'w executing	5,369,621	0.194957	0:02:18.579	00008	14	0.001719	select o_orderprio
1,405,114	WLMD8	oltp.exe	3	ADMINMM	UO'w executing	3,988,145	0.153462	0:02:20.159	00007	10	0.001284	select o_orderprio
2,695,574	WLMD8	oltp.exe	3	ADMINMM	UO'w executing	3,825,545	0.142983	0:02:12.511	00007	8	0.000619	select * from lpcd.o
1,155,466	WLMD8	oltp.exe	3	ADMINMM	UO'w executing	3,758,756	0.160713	0:01:57.850	00007	8	0.000994	select * from lpcd.o
1,003,880	WLMD8	oltp.exe	3	ADMINMM	UO'w executing	3,806,062	0.118867	0:01:41.988	00007	8	0.000670	select * from lpcd.o
2,531,522	WLMD8	oltp.exe	3	ADMINMM	UO'w executing	3,763,398	0.128274	0:01:25.341	00007	8	0.000395	select * from lpcd.o
2,179,316	WLMD8	oltp.exe	3	ADMINMM	UO'w executing	3,567,772	0.173766	0:01:21.688	00007	10	0.000034	select o_orderkey, c
2,236,432	WLMD8	oltp.exe	3	ADMINMM	UO'w executing	2,428,809	0.240179	0:01:21.150	00007	0	0.000000	select o_orderkey, c
2,273,708	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	6,668,065	0.291896	0:00:09.000	00005	6	0.000324	select l_orderkey, s
1,302,896	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	12,984,749	0.660977	41:83:95.74	00006	4	0.001312	select o_year, sum#
1,184,188	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	6,913,336	0.749968	9:20:40.000	00007	6	0.000034	select s_name, c_c
2,457,760	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	35,972,127	0.596748	0:01:32.433	00010	48	0.003972	select s_name, cou
2,281,510	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	14,899,738	0.368999	0:02:03.409	00007	16	0.000084	select c_name, c_c
1,778,018	WLMD8	dsx.exe	5	DSSUSER	UO'w executing	35,924,274	0.580834	0:01:26.335	00010	48	0.003953	select s_name, cou
1,196,528	WLMD8	dsx.exe	5	DSSUSER	UO'w executing	13,176,095	0.474151	43:70:76.13	00006	4	0.001333	select o_year, sum#
1,573,070	WLMD8	dsx.exe	5	DSSUSER	UO'w executing	14,972,173	0.277497	0:01:57.399	00007	16	0.000091	select c_name, c_c
1,441,816	WLMD8	dsx.exe	5	DSSUSER	UO'w executing	6,596,560	0.274933	0:00:10.056	00005	6	0.000109	select l_orderkey, s
2,371,620	WLMD8	dsx.exe	5	DSSUSER	UO'w executing	6,831,413	0.661100	7:10:34.34	00007	6	0.000032	select s_name, s_a
2,084,942	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	5,799,922	0.708044	22:34:08.12	00007	5	0.000041	select s_name, s_a
1,597,480	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	5,266,834	0.170086	0:00:10.009	00005	0	0.000000	select o_year, sum#
499,864	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	8,448,539	0.170254	0:01:22.516	00006	8	0.000062	select c_name, c_c
1,556,678	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	3,982,979	0.715998	5:22:89.92	00006	12	0.000191	select s_name, s_a
1,708,134	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	5,988,759	0.092495	0:00:09.968	00005	3	0.002408	select l_returnflag, l
1,544,322	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	5,104,448	0.117321	0:00:08.000	00005	3	0.000002	select n_name, sur
2,261,144	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	4,611,015	0.103736	0:00:08.992	00005	3	0.000006	select o_orderprio
1,335,748	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	4,857,360	0.094240	0:00:36.653	00005	0	0.000000	select c_custkey, c
1,253,678	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	6,134,720	0.455697	51:35:09.000	00006	11	0.000001	select n_name, sur
860,494	WLMD8	dsx.exe	5	ADMINMM	UO'w executing	6,424,900	0.131198	0:00:09.968	00005	6	0.000404	select l_orderkey, s

Figure 11-23 Application Summary - sorted by Workload ID

Viewing workload statistics and histograms

In Figure 11-24, we can see a summary view of all the most recently captured WLM statistics data.

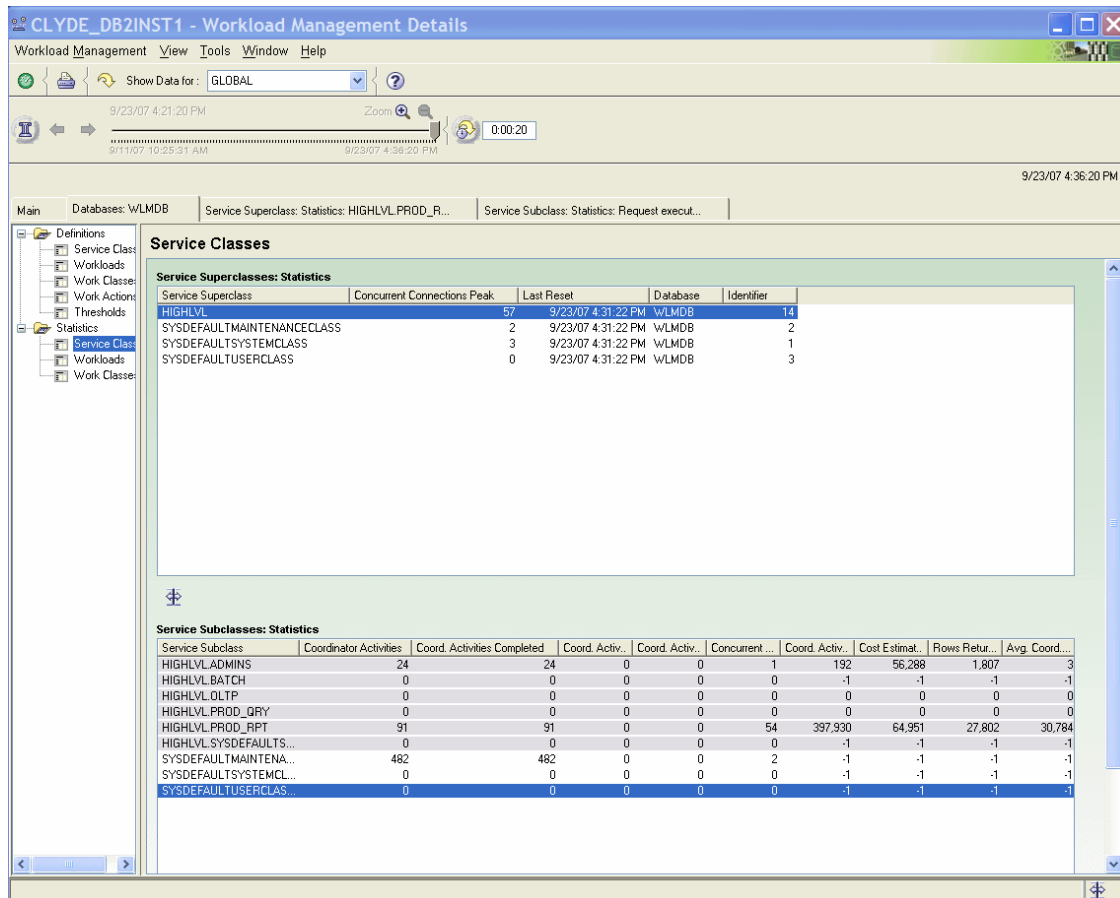


Figure 11-24 WLM Service Class statistics - summary view

The PROD_RPT service class is the only one with statistics at the moment, so we would like to drill down to see more information about that one. We double-click on the HIGHLVL.PROD_RPT subclass on the lower part of the window. This opens up another tab, as we see in Figure 11-25. This is more or less the same information as on the summary page, but you can view it for just one subclass. The lower area of the window references some Histogram statistics that are available.

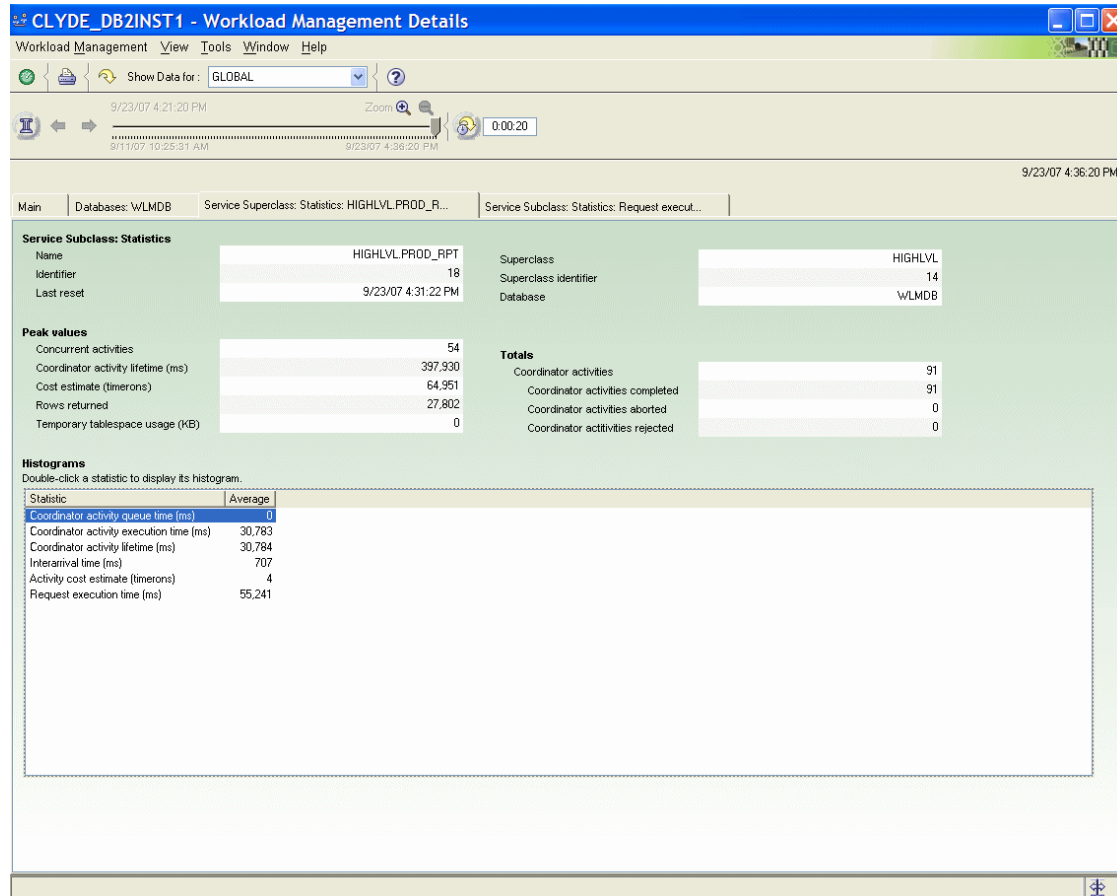


Figure 11-25 WLM Subclass statistics - HIGHLVL.PROD_RPT

We double-click the Request execution time (ms) statistic, to open up another tab where we can view the histogram chart, which is shown in Figure 11-26.

The analysis and conclusions about the performance data are the same as were described in earlier sections of this book. The benefit of using DB2 Performance Expert is that you can get at the data more quickly.

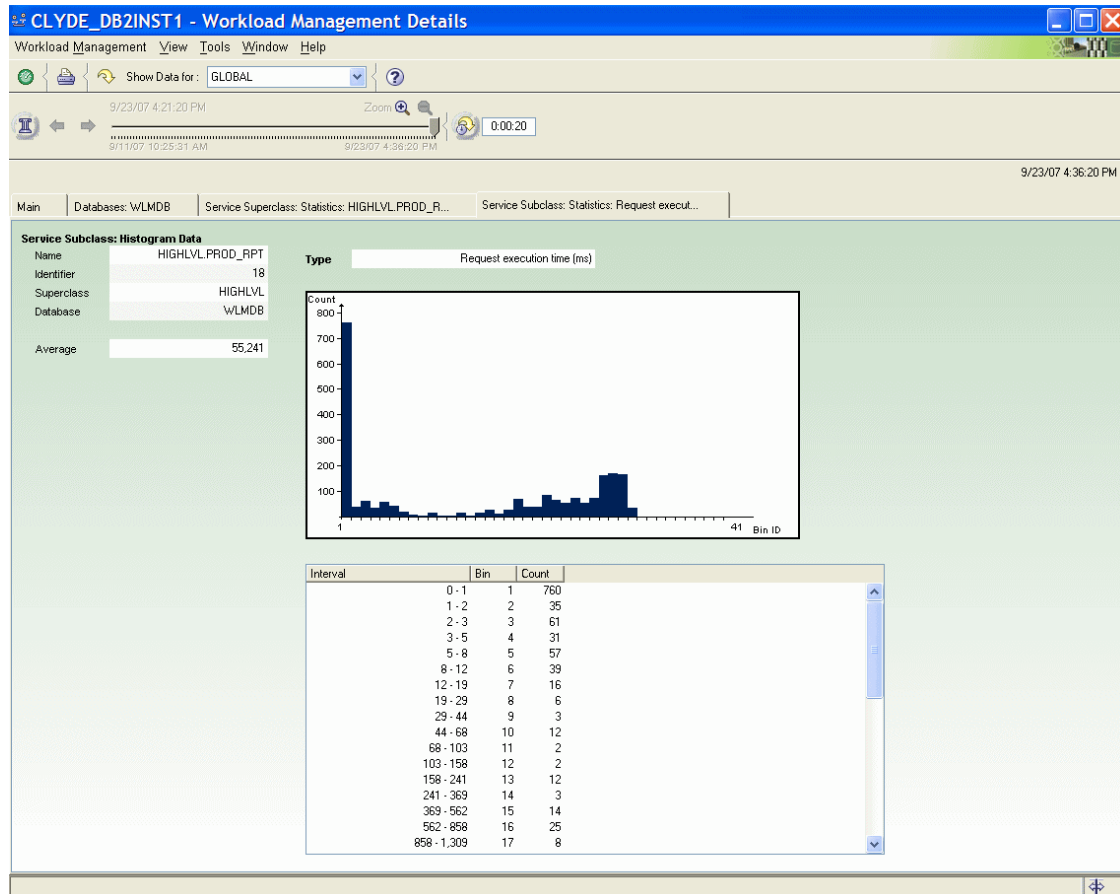


Figure 11-26 WLM Histogram view for PROD_RPT Request execution time

Viewing long-term WLM statistics through PE GUI

The DB2 Performance Expert server captures DB2 performance statistics using the snapshot facility, and WLM performance statistics using the event monitor. The performance data is stored in DB2 tables in the PE performance database.

The detailed short-term history data is what you see on the screens in the PE GUI when you are in history mode, and is the primary way in which you access the short-term data.

The short-term data is automatically aggregated and stored in different tables in the PE performance database. These tables are what comprise the Performance Warehouse (PWH). Traditionally, the PWH data is only accessible through the reports and queries provided by PE in the PWH screens. With more recent

versions of PE, however, you can view many of the DB2 and OS counters from the GUI screen, in the form of a trend analysis graph.

To access the PWH trend analysis, you must right-click on the performance counter you are interested in, to bring up the context menu. Not all performance counters can be viewed this way, so if the context menu item is disabled (grayed out) the counter is not available. If it is not grayed out, select the item, as shown in Figure 11-27. In this case we are looking at a WLM statistic - the peak value for cost estimate (timerons) for the HIGHLVL.ADMINS service class.

The screenshot shows the 'CLYDE_DB2INST1 - Workload Management Details' window. The main content area displays statistics for the service class 'HIGHLVL.ADMINS'. A context menu is open over the 'Cost estimate (timerons)' value of 56,288, with the option 'Analyze Performance Warehouse History' highlighted. Below the statistics is a 'Histograms' section with a table of statistics and their averages.

Statistic	Average
Coordinator activity queue time (ms)	0
Coordinator activity execution time (ms)	450
Coordinator activity lifetime (ms)	459
Interarrival time (ms)	N/P
Activity cost estimate (timerons)	N/P
Request execution time (ms)	N/P

Figure 11-27 Launching the PWH trend analysis

When you launch the PWH trend analysis, a new tab opens and a graph is displayed. The graph shows the actual values (in blue), but also calculates a historical trend (dark gray) based on those values, and a future projection (light

gray) of the values. In Figure 11-28, we see the trend for the cost estimate WLM counter. You can adjust the view to show longer or shorter time ranges up to one year.

These trend charts can be helpful to quickly spot when something may be either trending out of good performance or maybe a temporary spike has occurred that you can investigate further. Along with the WLM counters we see here, the trend charts are available for OS and DB2 statistics counters such as paging space usage, or buffer pool hit ratios, table pages used, rows read and so on.

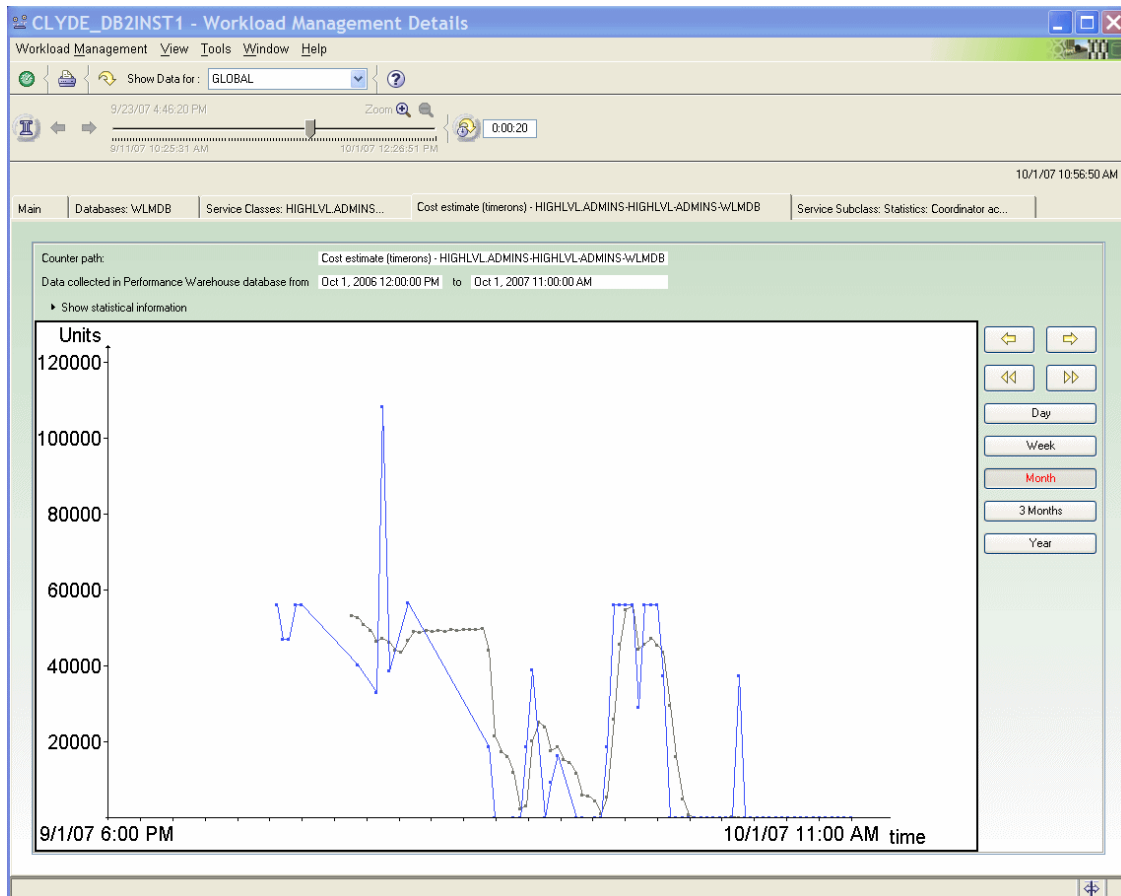


Figure 11-28 PWH Trend Analysis chart - timerons

Viewing WLM statistics with PE Performance Warehouse

When you want to find more detail about long-term performance data, you can use the predefined queries and reports that are in the Performance Warehouse

(PWH). You can also create your own queries or modify the ones that come with PE.

In Figure 11-29, we see a list of the predefined queries that come with PE. We are interested in the WLM-related queries. In this example we execute the query for WLM Workload Definitions.

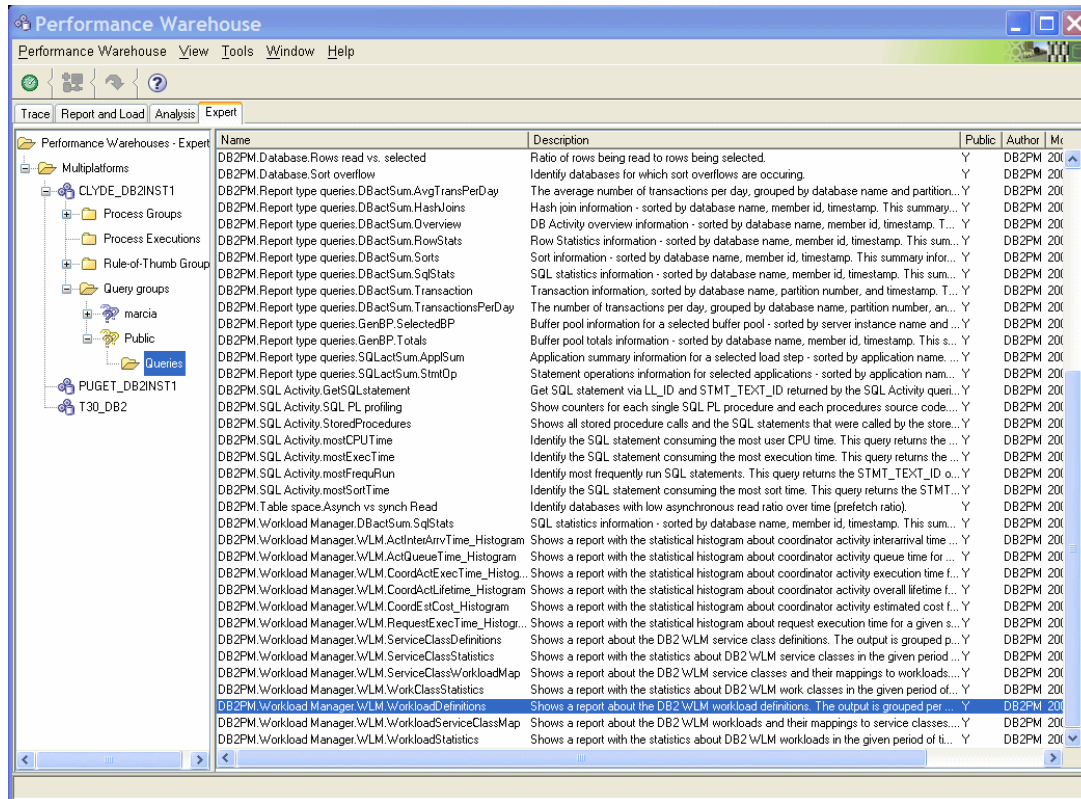


Figure 11-29 PE Performance Warehouse - Predefined Queries

To execute a query as-is, you can just right-click the query and select EXECUTE from the context menu. In many cases you might like to modify the query a bit to suit your own needs, but we do not explore that in this section.

Note: To see more examples about how to use the DB2 Performance Expert Performance Warehouse queries and reports, see IBM Redbooks publication, *DB2 Performance Expert for Multiplatforms V2.2*, SG24-6470.

After executing the query, the results are displayed in the PE window as shown in Figure 11-30. You can save the results to a text file, or view them in a browser, where you could also save the HTML output.

Query Execution

jdbc:db2:CETUSPE - Public - DB2PM\Workload Manager\WLM\WorkloadDefinitions

View SQL View Result

```

EVALUATIONORDER, ENABLED, EXTERNALNAME, ALLOWACCESS, COLLECTAGGACTDATA,
COLLECTACTA, COLLECTACTPARTITION, REMARKS
FROM PwHL_8\WORKLOADS\WL_WLCON_FINAL
WHERE WL_INTERVAL_TO=WLCON_FINAL.INTERVAL_TO AND
WL_WORKLOADID=WLCON_FINAL.WORKLOADID AND
WL_DB_NAME=WLCON_FINAL.DB_NAME
ORDER BY WL_DB_NAME, WL_WORKLOADNAME, CREATE_TIME, ALTER_TIME

```

DB_NAME	WORKLOADNAME	CREATE_TIME	ALTER_TIME	CONNECTOR_ATTRIBUTES	EVALUATIONORDER	ENABLED	ALLOWACCESS	COLLECTAGGACTDATA	COLLECTACT
WLMDB	WL_PROD_QRY	2007-09-10 14:21:26.620597	2007-09-19 14:33:45.644374	SESSION_USER GROUP=DSSGROUP	7	N	Y	N	N
WLMDB	WL_PROD_QRY	2007-09-21 17:34:42.656457	2007-09-21 17:34:47.54328	SESSION_USER GROUP=DSSGROUP	3	Y	Y	N	N
WLMDB	WL_PROD_QRY	2007-09-22 17:27:09.717608	2007-09-22 17:27:14.438634	SESSION_USER GROUP=DSSGROUP	4	Y	Y	N	N
WLMDB	WL_PROD_QRY	2007-09-22 17:27:09.717608	2007-09-22 17:27:14.438634	SESSION_USER GROUP=DSSGROUP	7	Y	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-06 17:59:11.872145	2007-09-06 17:59:11.992007	APPLNAME=dds.exe	3	Y	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-10 11:57:41.26676	2007-09-10 11:57:41.26676	APPLNAME=dds.exe	3	N	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-10 14:21:26.614673	2007-09-10 14:21:26.746295	APPLNAME=dds.exe	3	Y	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-10 14:21:26.614673	2007-09-11 21:16:58.680382	APPLNAME=dds.exe	3	N	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-10 14:21:26.614673	2007-09-11 22:09:18.991855	APPLNAME=dds.exe	3	N	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-10 14:21:26.614673	2007-09-11 22:09:18.991855	APPLNAME=dds.exe	6	N	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-10 14:21:26.614673	2007-09-12 00:36:42.537518	APPLNAME=dds.exe	6	Y	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-10 14:21:26.614673	2007-09-12 20:08:30.762667	APPLNAME=dds.exe	6	N	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-10 14:21:26.614673	2007-09-13 18:30:57.775947	APPLNAME=dds.exe	6	Y	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-10 14:21:26.614673	2007-09-14 14:28:06.705287	APPLNAME=dds.exe	6	N	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-10 14:21:26.614673	2007-09-14 16:39:00.4224	APPLNAME=dds.exe	6	N	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-10 14:21:26.614673	2007-09-14 20:30:28.274156	APPLNAME=dds.exe	6	Y	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-10 14:21:26.614673	2007-09-19 14:33:45.638368	APPLNAME=dds.exe	6	N	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-21 17:34:42.364403	2007-09-21 17:34:47.242138	APPLNAME=dds.exe	2	Y	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-22 17:27:09.422116	2007-09-22 17:27:14.144052	APPLNAME=dds.exe	3	Y	Y	N	N
WLMDB	WL_PROD_RPT	2007-09-22 17:27:09.422116	2007-09-22 17:27:14.144052	APPLNAME=dds.exe	6	Y	Y	N	N
WLMDB	WL_SALES	2007-09-11 22:46:55.42178	2007-09-11 22:46:55.534731	CURRENT_CLIENT_APLNAME=oltp.exe, ...	3	Y	Y	N	N
WLMDB	WL_SALES	2007-09-11 23:43:02.8022	2007-09-12 00:09:49.38275	CURRENT_CLIENT_APLNAME=oltp.exe, ...	3	Y	Y	N	D
WLMDB	WL_SALES	2007-09-11 23:43:02.8022	2007-09-12 00:34:05.204766	CURRENT_CLIENT_APLNAME=oltp.exe, ...	3	N	Y	N	D
WLMDB	WL_SALES	2007-09-11 23:43:02.8022	2007-09-12 20:07:49.519766	CURRENT_CLIENT_APLNAME=oltp.exe, ...	3	Y	Y	N	D
WLMDB	WL_SALES	2007-09-12 20:41:57.346407	2007-09-12 20:42:13.314549	APPLNAME=oltp.exe	1	Y	Y	N	N
WLMDB	WL_SALES	2007-09-12 21:18:20.142967	2007-09-12 21:18:35.570284	APPLNAME=oltp.exe	1	Y	Y	N	N
WLMDB	WL_SALES	2007-09-12 21:18:20.142967	2007-09-13 18:31:52.242329	APPLNAME=oltp.exe	1	N	Y	N	N
WLMDB	WL_SALES	2007-09-12 21:18:20.142967	2007-09-14 16:28:20.211118	APPLNAME=oltp.exe	1	N	Y	N	N
WLMDB	WL_SALES	2007-09-12 21:18:20.142967	2007-09-14 20:22:20.346444	APPLNAME=oltp.exe	1	Y	Y	N	N
WLMDB	WL_SALES	2007-09-12 21:18:20.142967	2007-09-19 14:33:21.804827	APPLNAME=oltp.exe	1	Y	Y	N	N

Row(s) 1 - 102 of 102

Save... Browse

Close

Figure 11-30 Performance Warehouse query results - Workload Definition

11.4 DB2 Performance Expert technical information

In this section, we discuss a few of the key technical and architectural points that can help you better understand how PE and WLM work together, and how WLM data collection is different from the DB2 snapshot performance data.

Database Configuration parameter WLM_COLLECT_INT

If you want to use PE to capture the WLM statistics information, you must set the database configuration parameter WLM_COLLECT_INT to 0. If you set WLM_COLLECT_INT to some other value, and you enable WLM monitoring for PE, then PE will reset it back to 0, because PE manages its own collection.

In a DPF environment, be sure to set this parameter on all partitions.

Event monitor naming convention

DB2 Performance Expert uses the STATISTICS event monitor to capture WLM performance data. PE will create the event monitor in the monitored database.

The event monitor name will be derived from the name of the PE server host and instance names. You will be able to see the name of the event monitor easily in the DB2 PE Application Summary window, because of how DB2 9.5 exposes the name in the snapshot, as shown in Figure 11-31. In this case the PE Server hostname is CETUS and the PE instance name is PEINST, so the event monitor name is CETUS__PEINST (though the full name is truncated in the snapshot output).

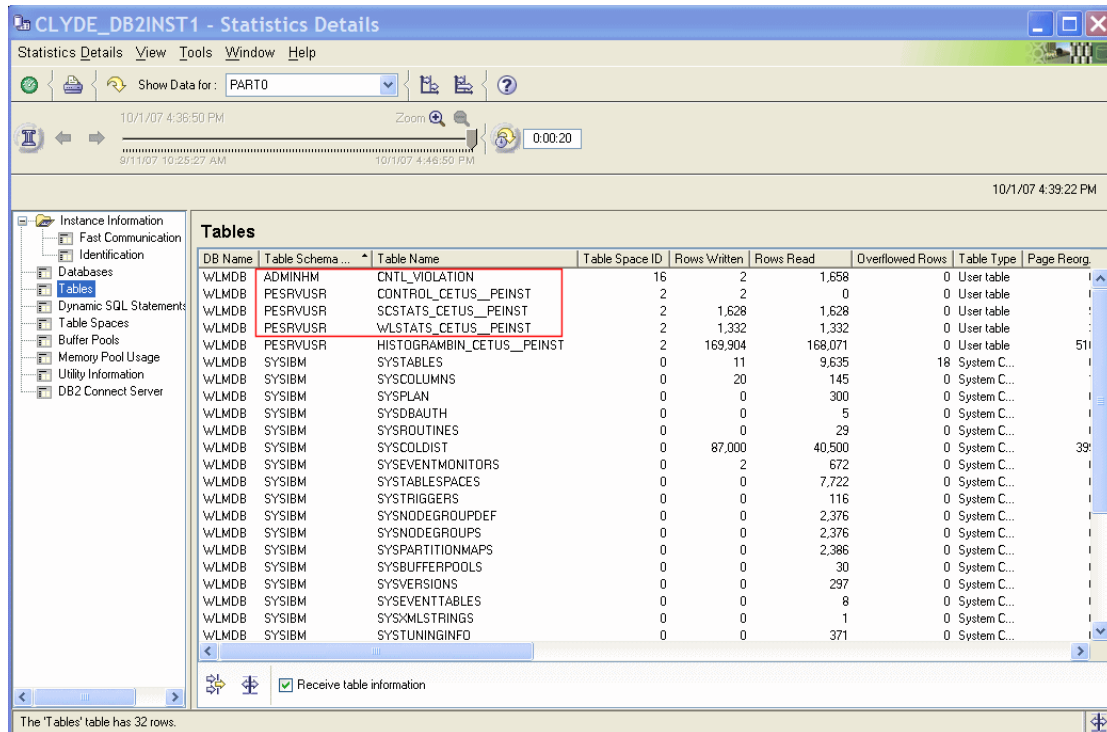
PID	DB Name	Application Name	Workload ID	Auth ID	Application Status	User CPU Time...	System CPU Time...	Host Execution Elap...	Seq#	Total Sorts	Total Sort Time (sec)
4.896	WLMDB	db2wlmnd	0	ADMINMM	connect completed	2.724150	5.632802	0.000000	00001	0	0.000000
4.896	WLMDB	db2bp.exe	7	ADMINMM	UOW waiting	0.011887	0.166021	0.000000	00002	0	0.000000
4.896	WLMDB	db2evmt_VIOLATIONS	0	ADMINMM	connect completed	2.838438	5.756795	0.000000	00001	0	0.000000
4.896	WLMDB	db2taskd	0	ADMINMM	connect completed	0.002957	0.000000	0.000000	00001	0	0.000000
4.896	WLMDB	db2evmt_CETUS__PEINS	0	PESRVUSR	connect completed	4.809842	5.759934	0.000000	00001	0	0.000000

Figure 11-31 PE Application Summary showing active event monitor

Notice that there is another event monitor active on this instance - called VIOLATIONS. This is also a WLM-related event monitor but it is not for Statistics, it is for the workload threshold violations so it is of the Activity type. The PE event monitor can coexist with activity event monitors you may create manually.

Event monitor tables

The PE Statistics event monitor captures the event data into tables in the monitored database. The tables are created under the schema of the user ID the PE server uses to connect to the monitored database. The user ID is specified during the PE server configuration. The table names themselves will carry the event monitor name along with the table-type as a prefix, for example see Figure 11-32.



DB Name	Table Schema	Table Name	Table Space ID	Rows Written	Rows Read	Overflowed Rows	Table Type	Page Reorg
WLMDB	ADMINHM	CNTL_VIOLATION	16	2	1,658	0	User table	
WLMDB	PESRVUSR	CONTROL_CETUS_PEINST	2	2	0	0	User table	
WLMDB	PESRVUSR	SCSTATS_CETUS_PEINST	2	1,628	1,628	0	User table	
WLMDB	PESRVUSR	WLSTATS_CETUS_PEINST	2	1,332	1,332	0	User table	
WLMDB	PESRVUSR	HISTOGRAMBIN_CETUS_PEINST	2	169,904	168,071	0	User table	511
WLMDB	SYSIBM	SYSTABLES	0	11	9,635	18	System C...	
WLMDB	SYSIBM	SYSCOLUMNS	0	20	145	0	System C...	
WLMDB	SYSIBM	SYSPLAN	0	0	300	0	System C...	
WLMDB	SYSIBM	SYSDBAUTH	0	0	5	0	System C...	
WLMDB	SYSIBM	SYSROUTINES	0	0	29	0	System C...	
WLMDB	SYSIBM	SYSCOLDIST	0	87,000	40,500	0	System C...	39
WLMDB	SYSIBM	SYSSEVENTMONITORS	0	2	672	0	System C...	
WLMDB	SYSIBM	SYSTABLESPACES	0	0	7,722	0	System C...	
WLMDB	SYSIBM	SYSTRIGGERS	0	0	116	0	System C...	
WLMDB	SYSIBM	SYSNODEGROUPDEF	0	0	2,376	0	System C...	
WLMDB	SYSIBM	SYSNODEGROUPS	0	0	2,376	0	System C...	
WLMDB	SYSIBM	SYSPARTITIONMAPS	0	0	2,386	0	System C...	
WLMDB	SYSIBM	SYSBUFFERPOOLS	0	0	30	0	System C...	
WLMDB	SYSIBM	SYSVERSIONS	0	0	297	0	System C...	
WLMDB	SYSIBM	SYSSEVENTABLES	0	0	8	0	System C...	
WLMDB	SYSIBM	SYSXMLSTRINGS	0	0	1	0	System C...	
WLMDB	SYSIBM	SYSTUNINGINFO	0	0	371	0	System C...	

Figure 11-32 Statistics Details - Tables view of PE event monitor tables

Event monitor activation and management

The PE server will retrieve the event monitor data that has accumulated during the collection interval you specify. When PE retrieves this data, it stores it in its own performance database on the PE server, and deletes it from the event monitor tables on the monitored database. The benefit of this approach is to keep the event monitor table size to a minimum on the monitored database, thus preserving your disk space.

If you create and enable event monitors outside of PE, you are responsible for ensuring the event monitor tables do not grow without bound. You can check this quite easily in PE, by looking at the Tables information in Statistics Details, as shown in Figure 11-33. In this example we look at the table `BASIC_MON_HISTOGRAMS`, which was used for some examples in earlier sections of this book when not discussing PE.

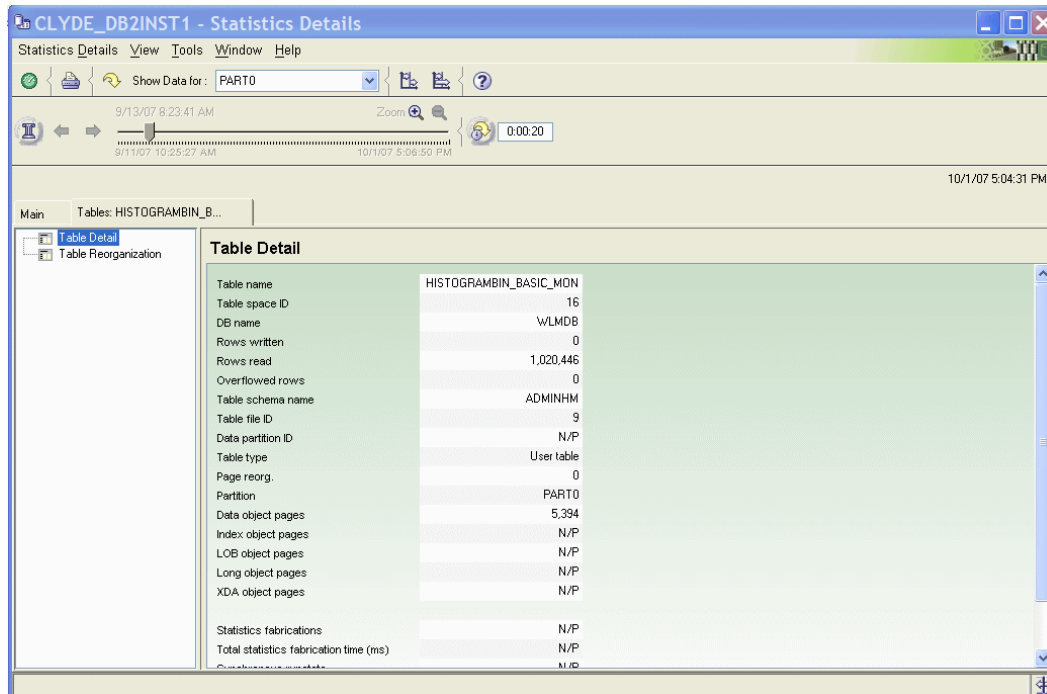


Figure 11-33 PE Table detail for non-PE histogram event monitor table

In the Redbook lab, we intentionally did not clear out this tables because we were conducting tests, but if you look at Figure 11-34, which is a PWH trend chart of the Data Object Pages counter, you can see how the table has grown over time and would continue to do so unless you took action.

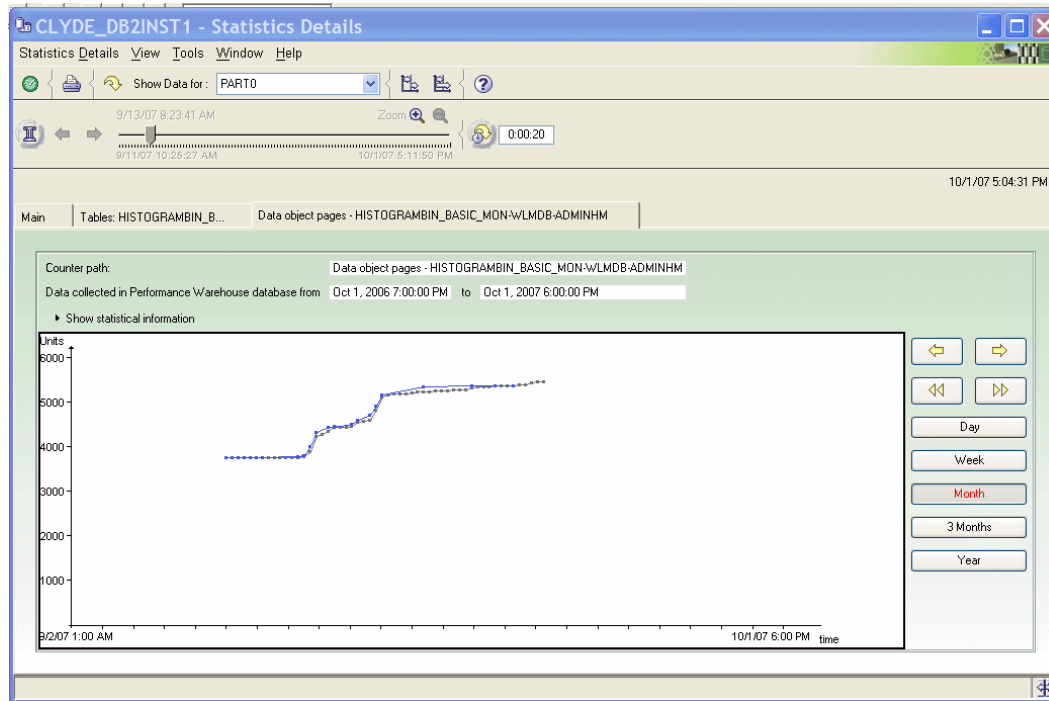


Figure 11-34 Trend chart of manual event monitor table size

Compare this to the PE event monitor table size, as shown in Figure 11-35, where you can see the table is much less volatile size.

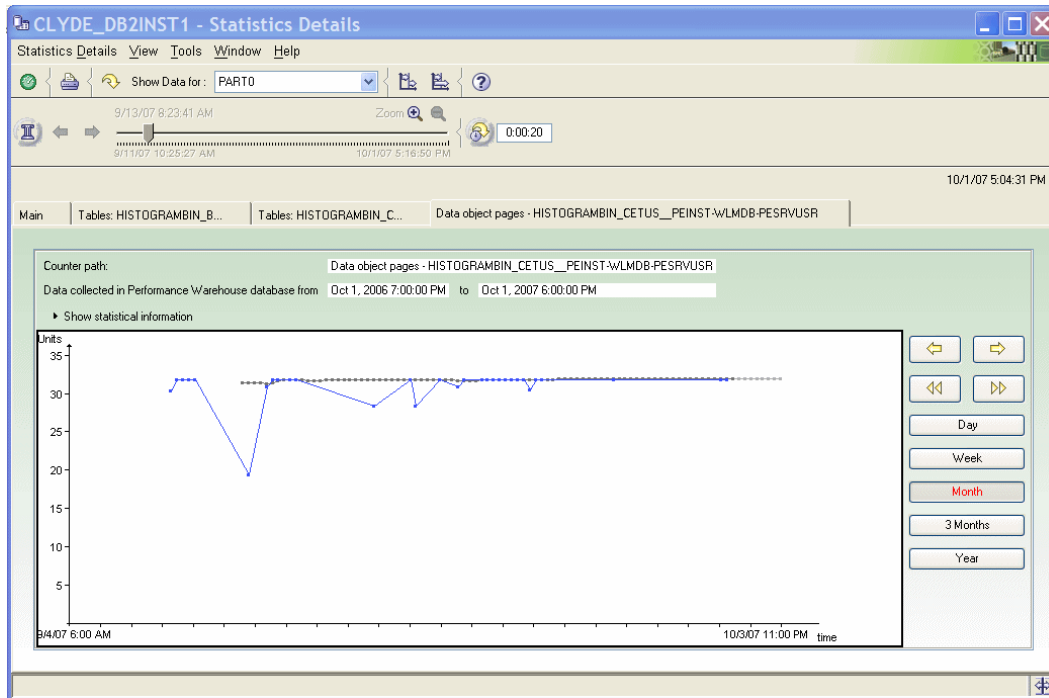


Figure 11-35 Trend chart of PE event monitor table size

PE will create and activate the event monitor when you enable history collection for WLM. This is enabled by default, so by default the event monitor and the associated tables are created when you start the PE server.

WLM collection interval

We mentioned above that PE does not use the `WLM_COLLECT_INT` parameter to control its WLM statistics collection. Instead, you should set the interval in the PE properties for the monitored instance as shown in Figure 11-36. You can set the collection time for the WLM definitions and the WLM statistics. To make reporting easier, it is most convenient to make the intervals the same. PE will check the definitions from the DB2 catalog, and flush the event monitor data at the interval you specify.

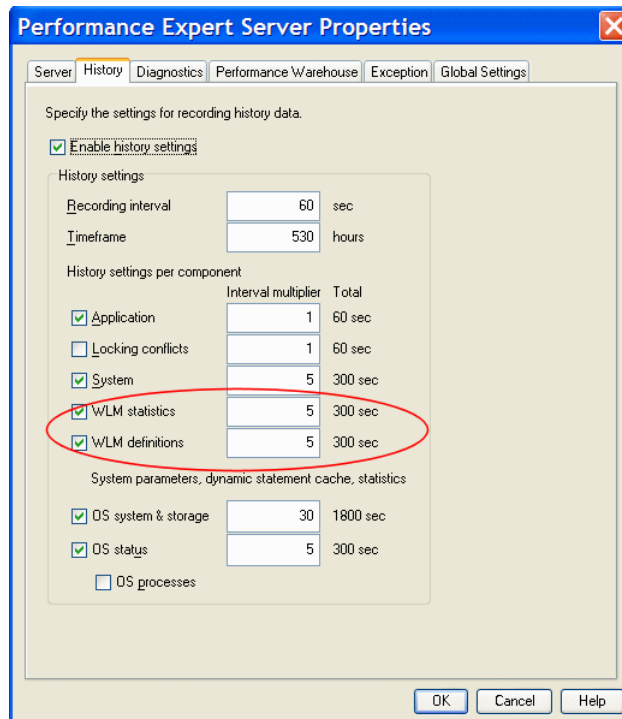


Figure 11-36 PE collection interval properties

The rest of the DB2 performance data is collected using snapshots, which are controlled by the other interval values on the properties page. These are independent from the WLM collection times.

Difference between WLM and DB2 statistics refresh

On all PE screens except WLM, when you refresh the screen you are asking the PE server to take a snapshot from the monitored database and show you the results. On the WLM screens, however, since it does not use snapshot, when you are using the GUI screens you are only ever looking at the most recent event monitor data collected by the PE server. Depending on the collection interval you specified, this could be fairly recent or could be older data. You can always know the date/time of the data on the screen by looking at the upper-right portion of the screen. This is true for all PE screens, not just WLM data.

If you are already familiar with PE's other features, this WLM behavior will seem quite strange. If you consider, however, the underlying architecture of how PE captures, stores, and presents the information to you in the GUI, you will get more comfortable with PE.

Coexisting event monitors

As we saw in the earlier chapters of this book, you can create your own event monitors for capturing WLM data. There are two types of event monitors - statistics and activities. PE currently only uses the statistics event monitor, so there are no coexistence issues with creating your own activity event monitor.

If you create and activate your own statistics event monitor - which also assumes you will modify the WLM_COLLECT_INT parameter - PE will not collect the WLM data. Your manual event monitor will supersede the PE event monitor. It is best to avoid this situation altogether and just let PE do the work for collecting WLM statistics data. If you do have another statistics event monitor, PE will indicate this on the WLM information page, as shown in Figure 11-37, and tell you to drop it. It does not really hurt anything to keep both defined, but your results may be unpredictable if you try to activate both of them.

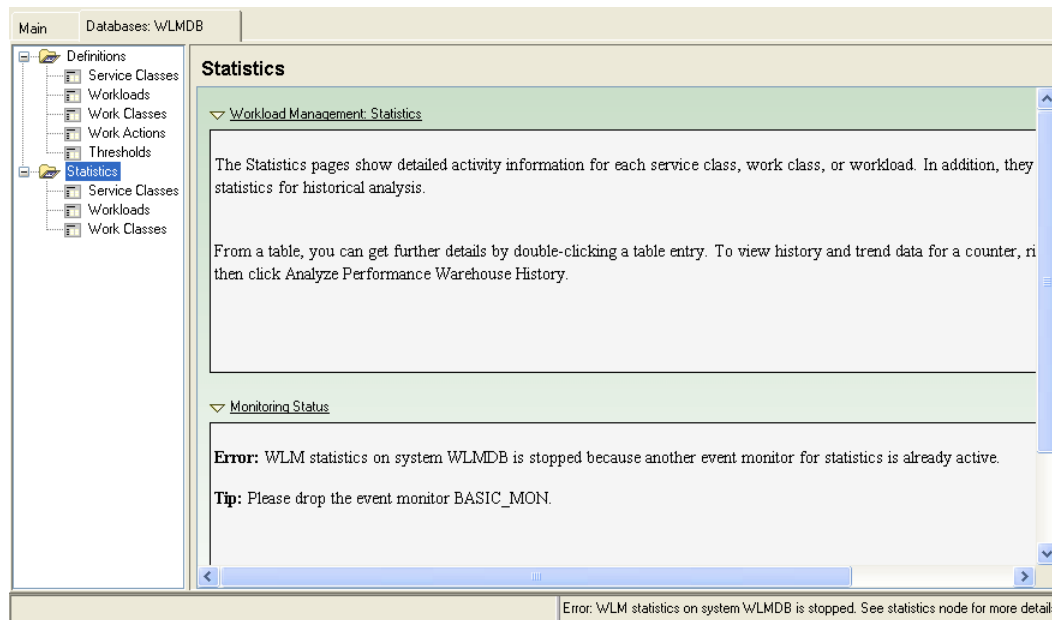


Figure 11-37 PE warning for duplicate event monitor



Administration

This chapter discusses the key points to check in administering a DB2 workload management environment. As you gain familiarity with how DB2 WLM works, you will be creating a number of workload management objects, and collecting baseline data and statistics for analysis. After enabling the WLM controls and fine-tuning the WLM monitoring, you will have a working WLM environment that meets your needs. Once migrating all of your WLM settings to full production, you need to periodically purge old data, backup the WLM settings, and take note of the tools needed in problem diagnosis.

We discuss how to administer a WLM environment through the following topics:

- ▶ WLM logs and maintenance
- ▶ WLM problem diagnosis
- ▶ WLM backup and recovery
- ▶ WLM authorization

12.1 WLM logs and maintenance

All WLM object definitions, functions, and data are stored in the WLM-specific DB2 catalog tables and the DB2 event monitor tables. To determine the WLM-specific messages being written out, you should check the DB2 diagnostic log, db2diag.log, and the administration notification log, named <DB2 instance name>.nfy.

PD_GET_DIAG_HIST table function

In DB2 9.5, a new SQL administrative table function for logging facilities called PD_GET_DIAG_HIST is available. This table function can return event notification and diagnostic log records from many facilities such as DB2 workload management, optimizer statistics, and the administration notification logs.

The information returned by the PD_GET_DIAG_HIST table function is listed as follows:

- ▶ FACILITY - A logical grouping which records relate to
- ▶ RECTYPE - The type of record
- ▶ TIMESTAMP - The time that message was created
- ▶ TIMEZONE - The time difference in minutes from Universal Coordinated Time (UTC)
- ▶ INSTANCENAME - The name of the instance where the message was created
- ▶ DBPARTITIONNUM - The partition number where the message was created
- ▶ LEVEL - The severity level of the record
- ▶ IMPACT - The impact of the message from a user's perspective
- ▶ DBNAME - The name of the database being accessed while this message was created
- ▶ EDU_ID - The engine dispatched unit identifier that created this message
- ▶ EDUNAME - The engine dispatched unit that created this message
- ▶ PID - The operating system process identifier that created this message
- ▶ PROCESS_NAME - The operating system process name that created this message
- ▶ TID - The thread numerical identifier that created this message
- ▶ APPLNAME - The name of the client application that initiated the connection, if it is available

- ▶ APPLHANDLE - A system-wide unique identifier for the application that initiated the application when available
- ▶ AUTH_ID - The system authorization identifier for the process
- ▶ PRODUCT - The name of the product that created the message
- ▶ COMPONENT - The name of the component that created the message
- ▶ FUNCTION - The name of the function that generated the message
- ▶ PROBE - The probe point number used to identify where the message was generated in the function
- ▶ CALLEDPRODUCT - The name of the product at the source of the error
- ▶ CALLEDCOMPONENT - The name of the component at the source of the error
- ▶ CALLEDFUNCTION - The name of the function at the source of the error
- ▶ OSERR - The operating system error number
- ▶ RETCODE - The product specific return code
- ▶ MSGNUM - The numeric message number for the associated message, if it is available
- ▶ MSGTYPE - The type related to the message identifier
- ▶ MSG - A short description text for this record
- ▶ OBJTYPE - This is the type of object the event applies to, if it is available. The values that are WLM-specific are:
 - HISTOGRAM TEMPLATE
 - SERVICE CLASS
 - THRESHOLD
 - WORK ACTION SET
 - WORK CLASS SET
 - WORKLOAD
- ▶ OBJNAME - The name of the object the event relates to, if it is available
- ▶ OBJNAME_QUALIFIER - This is additional information about the object.
- ▶ EVENTTYPE - The action or verb associated with this event
- ▶ EVENTDESC - A short representation of the key fields for this event
- ▶ FIRST_EVENTQUALIFIERTYPE - The type of the first event qualifier
- ▶ FIRST_EVENTQUALIFIER - The first qualifier for the event
- ▶ SECOND_EVENTQUALIFIERTYPE - The type of the second event qualifier
- ▶ SECOND_EVENTQUALIFIER - The second qualifier for the event
- ▶ THIRD_EVENTQUALIFIER_TYPE - The type of the third event qualifier

- ▶ THIRD_EVENTQUALIFIER - The third qualifier for the event
- ▶ EVENTSTATE - The state of the object or action as a result of the event
- ▶ EVENTATTRIBUTE - The event attributes
- ▶ EVENTSTACK - The logical event stack at the point the record was logged when applicable
- ▶ CALLSTACK - The operating system stack dump for the thread that generated this record when applicable
- ▶ DUMPFIL - The name of the secondary dump file associated with the log record when applicable
- ▶ FULLREC - The formatted text version of the entire record

WLM-specific records can be retrieved using this table function if the COMPONENT column is set to "WLM". The table function PD_GET_DIAG_HIST is invoked by passing the following parameters:

```
PD_GET_DIAG_HIST (facility, rectype, impact, start_time, end_time)
```

Example 12-1 invokes the PD_GET_DIAG_HIST table function with a RECTYPE EX and all WLM-specific records.

Example 12-1 Querying the PD_GET_DIAG_HIST table function

```
SELECT facility, rectype, timestamp, impact,
       SUBSTR(objtype,1,18) AS objtype,
       SUBSTR(msg,1,50) AS msg,
       fullrec
FROM TABLE (PD_GET_DIAG_HIST('MAIN','EX','',
                             CAST(NULL AS TIMESTAMP),
                             CAST(NULL AS TIMESTAMP))) AS T
WHERE component = 'WLM';
```

Example 12-2 shows part of the output results from the query.

Example 12-2 Sample output results from PD_GET_DIAG_HIST table function

```
MAIN          EX      2007-09-07-18.05.46.629491 Critical
SERVICE CLASS -
2007-09-07-18.05.46.629491-300 E10548172A501    LEVEL: Event
PID   : 721158                TID   : 14774    PROC  : db2sysc 0
INSTANCE: db2inst1           NODE  : 000      DB   : WLMDB
APPHDL : 0-248                APPID: *NO.db2inst1.070907230531
AUTHID  : DB2INST1
EDUID   : 14774                EDUNAME: db2agent (WLMDB) 0
FUNCTION: DB2 UDB, WLM, sqlrwCommitWLMDDL, probe:500
ALTER   : SERVICE CLASS : SYSDEFAULTSYSTEMCLASS : success
```

```

IMPACT : Critical
DATA #1 : signed integer, 4 bytes
1
MAIN          EX          2007-09-07-18.05.59.031956 Critical
SERVICE CLASS -
2007-09-07-18.05.59.031956-300 E10548674A506          LEVEL: Event
PID          : 721158          TID : 14774          PROC : db2sysc 0
INSTANCE: db2inst1          NODE : 000          DB : WLMDB
APPHDL      : 0-248          APPID: *NO.db2inst1.070907230531
AUTHID      : DB2INST1
EDUID       : 14774          EDUNAME: db2agent (WLMDB) 0
FUNCTION: DB2 UDB, WLM, sqlrwCommitWLMDDL, probe:500
ALTER       : SERVICE CLASS : SYSDEFAULTMAINTENANCECLASS : success
IMPACT      : Critical
DATA #1     : signed integer, 4 bytes
2

```

WLM maintenance

If you are using automatic collection of workload management statistics, the event monitor files and tables can be filled up over time. You need to prune your event monitor files or tables periodically to ensure that workload management statistics collection does not stop unexpectedly.

The workload management statistics table functions report the current values of the in-memory statistics. If you have automatic workload management statistics collection enabled, these values are reset periodically on the interval defined by the `WLM_COLLECT_INT` database configuration parameter. When looking at the statistics reported by the table functions, you should always consider the `LAST_RESET` column. This column indicates the last time the in-memory statistics were reset. If the time interval between the last reset time to the current time is not sufficiently large, there may not be enough data collected to allow you to draw any meaningful conclusions.

A reset of workload manager statistics applies to all users.

12.2 WLM problem diagnosis

When a problem is encountered in WLM, the symptoms of the problem must first be examined. The following questions can be asked to try to narrow down the source of the problem:

- ▶ Does the problem have anything to do with how a WLM object was defined?
WLM problems can usually result if the WLM object definitions are incorrectly created or altered. Review the WLM object definitions to ensure that the WLM

objects are valid. Using Design Studio is one way to help validate WLM object definitions before they are created or altered. If you have DB2 Performance Expert installed, you also can view the DB2 WLM definitions from PE.

- ▶ Does the problem in WLM occur by itself, or does it occur when other DB2 problems occur?

Review the diagnostic logs, db2diag.log, and the administration notification log, < DB2 instance name>.nfy to determine if the WLM problem is related to a DB2 problem and isolate the source.

- ▶ Is the problem related to a WLM workload or workload occurrence?

You need to gather the following information about workloads:

- Get the list of workload occurrences using the WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURENCES table function.
- Get the identifier for the workload and workload occurrence using the WLM_GET_WORKLOAD_OCCURENCE_ACTIVITIES table function.
- Get the list of all activities and requests running under a workload occurrence using the WLM_GET_WORKLOAD_OCCURENCE_ACTIVITIES table function.
- Get workload information in memory using **db2pd -db <database name> -workloads** command.
- Get information from the PD_GET_DIAG_HIST table function and the DB2 diagnostic logs.

- ▶ Is the problem related to a WLM service class using more than a fair share of agents?

One possible cause is a data server resource, such as an agent, is overutilized by a group of users or an application. A service class may be using more than its fair share of agents. Determine the service class that is overutilizing resources and take action to limit the resources being used by the service class.

Example 12-3 illustrates the use of the WLM_GET_SERVICE_CLASS_AGENTS table function to determine how many agents are working for each service class.

Example 12-3 Determine how many agents are working for each service class

```
SELECT SUBSTR(agents.service_superclass_name,1,19) AS
superclass_name,
       SUBSTR(agents.service_subclass_name,1,19) AS subclass_name,
       COUNT(*) AS agent_count
```

```

FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('', '', CAST(NULL AS
BIGINT), -2)) AS agents
WHERE agent_state = 'ACTIVE'
GROUP BY service_superclass_name, service_subclass_name
ORDER BY service_superclass_name, service_subclass_name

```

- ▶ Is the problem, that is, a system slowdown, related to the configuration of service classes?

Use the following principles to address this problem:

- Resolve DB2 locking conflicts on the application and environment if they exist.
- Increase the thresholds if a service class is running too close to its threshold levels.
- In an AIX environment, if the resources allotted to a DB2 service class are being exhausted, determine if the mapped AIX service classes are not getting sufficient CPU, I/O bandwidth, or other resources.
- An increased amount of activity in a service class could lead to abnormal resource consumption. Check the number of completed activities in the service class to determine if the amount of work being done in the service class is reasonable.
- More activities in a service class can lead to increased queue times for activities. Check to see if the average activity lifetime for a particular service class has increased. If the increase in average lifetime is unacceptable, allocate more resources to the service class and reduce the concurrency threshold. The query in Example 12-4 can be used to get a high-level overview of what is occurring on a service class. A sharp increase in ACTAVGLIFETIME over time could be an indication that the resources for a service class are being exhausted.

Example 12-4 Query to get a high-level overview of activity in a service class

```

SELECT SUBSTR(service_superclass_name,1,19) AS superclass_name,
       SUBSTR(service_subclass_name,1,18) AS subclass_name,
       SUBSTR(CHAR(SUM(coord_act_completed_total)),1,13) AS
         actscompleted,
       SUBSTR(CHAR(SUM(coord_act_aborted_total)),1,11) AS actsaborted,
       SUBSTR(CHAR(MAX(concurrent_act_top)),1,6) AS actshw,
       CAST(CASE WHEN SUM(coord_act_completed_total) = 0 THEN 0
             ELSE SUM(coord_act_completed_total *
                     coord_act_lifetime_avg) /
                     SUM(coord_act_completed_total)
             END / 1000 AS DECIMAL(9,3)) AS actavglifetime
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS ('', '', -2)) AS scstats
GROUP BY service_superclass_name, service_subclass_name

```

ORDER BY service_superclass_name, service_subclass_name

- ▶ Is the problem related to not seeing the expected behavior when using AIX WLM with DB2 WLM?

If you are not seeing the desired behavior, you may need to adjust the AIX WLM configuration. The AIX WLM configuration can be reviewed with the help of AIX WLM tools such as wlmstat, wlmmon. and wlmperf.

In summary, WLM problem diagnosis requires the use of DB2 diagnostic logs, the WLM table functions such as PD_GET_DIAG_HIST, WLM_GET_SERVICE_CLASS_AGENTS, WLM_GET_SERVICE_CLASS_STATS and the db2pd utility to get the needed information on what is happening in the system.

The db2pd utility

The db2pd utility is used to retrieve information from DB2 database system memory sets to help in problem diagnosis. In DB2 9.5, this utility has additional options to help get basic information about workloads, work action sets, and work class sets. The db2pd utility can be used to return the following information:

- ▶ A list of workload definitions in memory
- ▶ All enabled work action sets
- ▶ All enabled work actions in the enabled work action sets
- ▶ All work class sets referenced by an enabled work action set
- ▶ All work classes in those work class sets

Example 12-5 shows how to invoke the db2pd options for WLM.

Example 12-5 Using the db2pd utility to get WLM information

```
db2pd -db WLMDB -workloads -workactionsets -workclasssets
```

Example 12-6 is a sample portion of the output of the db2pd.

Example 12-6 Sample output from the db2pd command

```
Database Partition 0 -- Database WLMDB -- Active -- Up 0 days 05:23:56
```

Workload Definition:

Address	WorkloadID	WorkloadName	DBAccess
ConcWLOThresID	MaxConcWLOs	WLOActsThresID	MaxWLOActs
ServiceClassID			
0x07700000AB7B0080	3	WL_OLTP	ALLOW 0
9223372036854775806	0	9223372036854775806	20
0x07700000AB7B0168	4	WL_BATCH	ALLOW 0
9223372036854775806	0	9223372036854775806	17

0x07700000AB7B0250 5	WL_PROD_RPT	ALLOW	0
9223372036854775806 0	9223372036854775806	18	
0x07700000AB7B0340 6	WL_PROD_QRY	ALLOW	0
9223372036854775806 0	9223372036854775806	19	
0x07700000AB7B0430 7	WL_ADMIN	ALLOW	0
9223372036854775806 0	9223372036854775806	16	
0x07700000AB7B0518 1	SYSDEFAULTUSERWORKLOAD	ALLOW	0
9223372036854775806 0	9223372036854775806	13	
0x07700000AB7B05E8 2	SYSDEFAULTADMWORKLOAD	ALLOW	0
9223372036854775806 0	9223372036854775806	13	

...

Only the DB2 instance owner can run the db2pd utility. If you are not the DB2 instance owner, you will get the following error message:

```
Unable to attach to database manager on partition 0.
Please ensure the following are true:
- db2start has been run for the partition.
- db2pd is being run on the same physical machine as the
partition.
- DB2NODE environment variable setting is correct for the
partition
or db2pd -dbp setting is correct for the partition.
```

12.3 WLM backup and recovery

WLM information such as service class, workload, work action set, work class set and threshold objects are kept in the DB2 system catalog. Therefore, all WLM settings and vital WLM information are backed up automatically once the DB2 catalog table space is backed up.

However, to save the WLM object definitions, we recommend you run the **db2look** utility to backup the WLM definitions. The following command saves the WLM object definitions in a file:

```
db2look -d WLMDB -wlm -o wlm.definitions.out
```

This saves all the CREATE and ALTER statements for WLM objects, including all WLM CREATE event monitor statements.

12.4 WLM authorization

DB2 workload management objects do not have owners like regular DB2 database objects. For example, if a resource setting such as prefetch priority for a service class is changed, it affects not only the service class being changed, but other service classes of the same tier. The WLM administrator should have SYSADM or DBADM authority in order to manage WLM objects. If there are many WLM administrators within the organization, these administrators must frequently communicate with each other, otherwise they can inadvertently cancel or override WLM settings set by other administrators.

A session user must have the USAGE privilege on a workload before a workload can be associated with its DB2 connection. If the session user, or the group, or the role it belongs to does not have the USAGE privilege on a workload, then the data server will look at other matching workloads where that session user, or the group, or the role it belongs to, to see if it has the USAGE privilege. You can query the SYSCAT.WORKLOADAUTH catalog view to determine if a user, group, or role has been granted USAGE privilege on a workload.

The USAGE privilege on the SYSDEFAULTUSERWORKLOAD workload is granted to PUBLIC at database creation time, if the database was created without a RESTRICT option. If the database was created with the RESTRICT option, you must explicitly grant the USAGE privilege on the SYSDEFAULTUSERWORKLOAD workload to all non-SYSADM and non-DBADM users.

You have implicit USAGE privilege on all workloads if you have SYSADM or DBADM authority. The USAGE privilege on the SYSDEFAULTADMWORKLOAD workload cannot be granted, since it can only be used by SYSADM and DBADM users.

Example 12-7 shows the GRANT USAGE statement on the workload CAMPAIGN to user BILL, group SALES, and a ROLE called SALESPERSON. The USAGE privilege on the workload can also be granted to PUBLIC.

The GRANT USAGE statement takes effect when it is committed.

Example 12-7 Examples of the GRANT USAGE command

```
GRANT USAGE ON WORKLOAD campaign TO USER bill;  
GRANT USAGE ON WORKLOAD campaign TO GROUP sales;  
GRANT USAGE ON WORKLOAD campaign TO ROLE salesperson;  
GRANT USAGE ON WORKLOAD campaign TO PUBLIC;
```

You can use the REVOKE USAGE command to revoke USAGE from a user, group, or role from a workload. However, the REVOKE USAGE command does not work on the SYSDEFAULTADMWORKLOAD.

Example 12-8 shows how the USAGE privileges granted in Example 12-7 can be revoked.

Example 12-8 Example of the REVOKE USAGE command

```
REVOKE USAGE ON WORKLOAD campaign FROM USER bill;  
REVOKE USAGE ON WORKLOAD campaign FROM GROUP sales;  
REVOKE USAGE ON WORKLOAD CAMPAIGN FROM ROLE salesperson;  
REVOKE USAGE ON WORKLOAD campaign FROM PUBLIC;
```



Query Patroller and DB2 Governor

Prior to DB2 9.5, DB2 workload management consisted of Query Patroller (QP) and the DB2 Governor. Now that we have this new offering of the DB2 Workload Manager, it is quite natural to wonder why there was a need to build a separate offering. With the introduction of DB2 WLM, what happens to all the Query Patroller and Governor installations? How do I migrate a QP or Governor configuration to take advantage of WLM. We address these questions and more in this chapter.

We discuss the following topics:

- ▶ A brief background of the QP and Governor features.
- ▶ The differences between the workload management solutions of QP and Governor compared to what is available in DB2 9.5.
- ▶ An explanation of how DB2 WLM, QP, and the Governor can co-exist on the same data server.
- ▶ Some suggestions on how to approach the task of migrating from QP and the Governor to the new WLM features.

13.1 Query Patroller and DB2 Governor background

In this section, we give some background about Query Patroller and DB2 Governor. We also provide the difference between DB2 WLM, Query Patroller, and DB2 Governor.

13.1.1 Query Patroller

Query Patroller was designed as a predictive governing tool to manage query workloads based on who submitted the query and a cost-estimate (from the DB2 optimizer) representing a relative measure of resources required to execute the query. The primary management technique is to control the maximum concurrency of queries in these classifications. For example, to prevent a single user from monopolizing the data server resources, you could limit that user to running only 10 queries at a time and any queries over that threshold would be queued until one of the running queries completed. Even more useful is the concurrency control for queries classified by size. It is the very long queries that cause disruption to the shorter running queries, so it can be quite effective to limit the number of long running queries running at any given time.

Figure 13-1 on page 372 illustrates the Query Patroller environment

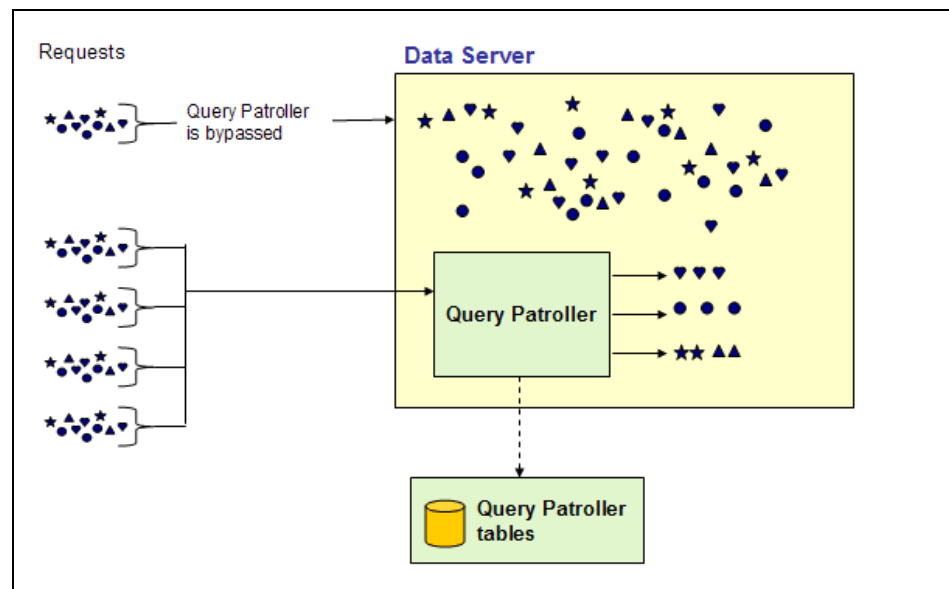


Figure 13-1 Query Patroller environment

In a QP environment, the life-cycle of a query looks something like the following:

- ▶ Queries requests are submitted to run on the server
- ▶ The query is intercepted on the server and QP determines if it is a query to be managed or bypassed. Managing queries requires some overhead, so it is often a good idea to minimize the overhead by selecting a set of queries, the really short queries or from a given application, to bypass QP altogether.
- ▶ Information about each managed query is written to a QP control table. The query attributes (submitter, estimated cost) are evaluated and a decision is made on whether to run the query now, queue the query until other work completes, or to put the query in a hold state (which is basically an error condition saying the query looks like it is just too large to run at all and it needs to be looked at).
- ▶ Once QP decides a query can be run, it is released to the data server to execute along with everything else on the system.
- ▶ QP is not aware of the query until the execution phase is complete and QP updates that query's control table entry with statistics about the execution results (for example, execution time, time queued, error code, and so on).

13.1.2 DB2 Governor

Query Patroller query management is fundamentally based on estimates and no matter how accurate those estimates may be, they are still estimates which means they could be wrong. Perhaps RUNSTATS has not been run for a while (the optimizer uses statistics to choose an optimal access plan, so they should be kept up to date), for example.

This is where the DB2 Governor nicely complements Query Patroller with its reactive approach to workload management. Once QP lets the query loose to execute, the DB2 Governor then watches for certain thresholds during the execution which can result in a couple of events to be triggered.

The thresholds include:

- ▶ Maximum execution time
- ▶ Maximum number of locks obtained and held
- ▶ Maximum number of rows returned to the application
- ▶ Maximum number of rows read while building a result set
- ▶ Maximum elapsed time since a unit of work became active
- ▶ Maximum connection idle time

The events that can be triggered include:

- ▶ Modify the agent priority at the operating system
- ▶ Force the application to terminate the connection and the database activity

13.1.3 Differences between QP, Governor, and WLM

While Query Patroller and the DB2 Governor are used successfully by many DB2 customers, as time goes on those customers are looking for more and more features that the existing products were not designed to address. DB2 WLM was built to address those needs in a flexible and scalable manner.

Table 13-1 highlights some of the main differences between workload management in DB2 9 (with QP and the Governor) and DB2 9.5 (with DB2 WLM). There are many other features in the products (such as concurrency control, or thresholds for elapsed time) that are found in both workload management offerings that are not listed in this table, but that is simply because the function is close to equivalent.

Table 13-1 Differences in Workload Management in DB2 9 and DB2 9.5

Query Patroller/ DB2 Governor	DB2 Workload Manager
Treats the whole database as its execution environment.	Allows multiple execution environments to be created using workloads and service classes.
Does not have any mechanism to explicitly control resources.	Provides mechanisms to explicitly control and influence resources during execution.
QP only manages DML activities and typically only the subsets of queries that have the biggest resource impact on the system (that is, shorter transactional queries are usually bypassed due to increased overhead).	All database activities are mapped to a service class. Many more database activities can be explicitly managed (e.g. DML, DDL, Load, and so on) The level of workload management is customized for each service class.
Once QP releases an activity for execution, it has no further influence (or information) on the activity until completion.	Keeps track of, and can continue to manage, activities throughout the life cycle of the work. Many monitoring functions are available to quickly determine the state of the workload.
Intercepts and keeps track of activities from the coordinator partition perspective.	Integrated into the engine, which allows for awareness and tracking of activities across partitions.
QP relies primarily on concurrency control based on query cost estimates for workload management	Explicit resource control for a specified execution environment based on estimated and real resource consumption.
Each managed query has detailed information recorded in a table.	Configurable as to the level of information to be captured for managed activities.

Query Patroller/ DB2 Governor	DB2 Workload Manager
QP handles predictive governing, based on estimates. The Governor handles reactive governing based on actual consumption.	Offers both predictive and reactive governing options.
When the Governor is set to stop an activity when a threshold is exceeded, the entire application connection is forced.	Threshold actions can be applied to a specific activity. As well as being able to cancel an activities, a less harsh option is available to simply capture details on the activity for future analysis.

13.2 Co-existing

Switching from a workload management solution using QP and the Governor to one using the WLM features has many benefits, but it is not feasible to assume that this can happen without some thought and a little bit of work. We discuss what the environment looks like in DB2 V9.5 and then discuss how to approach the task of moving to a WLM solution.

To dispel any misconceptions: QP and the Governor may not be the future strategic direction for workload management in DB2, but they are still fully supported in DB2 9.5 and are functionally equivalent to previous versions.

When you first install DB2 9.5, there is a default workload created to identify all the user database activities and map them to a default user service class. The default user service class is the execution environment where all database user activities will run. There are other database activities, like prefetching or some health monitoring activities, that would be mapped to either a system or maintenance service class. This discussion do not include those activities.

QP and the Governor only intercept and manage queries assigned and executing in the default user service class. In a vanilla install or migration, this includes all database activities so the life cycle of a query essentially stays the same except that before QP gets to take a look at the query, it is first assigned to the default service class and the Governor will only act on that same set of activities.

Figure 13-2 illustrates the Query Patroller in a default WLM environment.

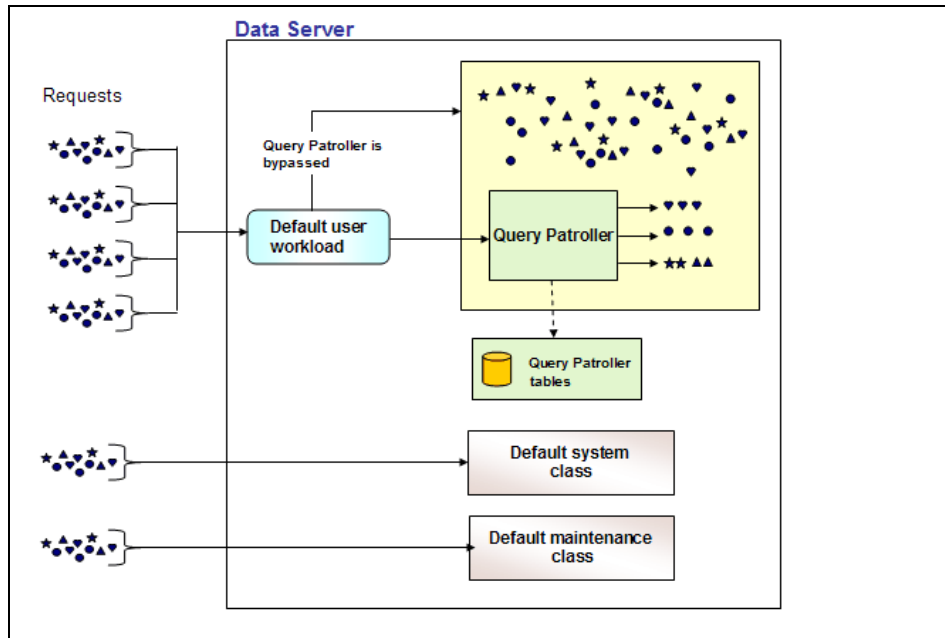


Figure 13-2 Query Patroller in a default WLM environment

If there are workloads defined to route user activities to service classes other than the default use class, Query Patroller and the Governor will not be able to manage the activities as they will bypass those tools completely.

13.3 Transitioning from Query Patroller and Governor

We just learned how QP and the Governor can, and will continue to, function properly on a data server at the version 9.5 level. In this section, we show you how to approach the transition of managing workloads with Query Patroller and the Governor to managing workloads with DB2 Workload Manager.

13.3.1 Is there a migration tool?

Many Query Patroller customers have achieved success with managing workloads using query classes and other submitter thresholds. A lot of time may have been spent build the configuration and they are comfortable with what they have built. It is only natural that these customers would hope for a migration tool that would simply map a QP configuration to a WLM configuration.

However, even though it is certainly possible to map a QP configuration to a WLM one, a migration tool is not provided. The reason is because the fundamental architecture of the two products is quite different. QP relies primarily on concurrency thresholds and query classes to control the maximum number of queries running at any given time, often based on query access plan cost estimates. DB2 WLM, on the other hand, has more concrete control options on actual resources in addition to concurrency controls.

A migration tool would have resulted in a fairly complex DB2 WLM configuration when it is quite likely that a more straight forward one would have done the same (and probably better) job.

13.3.2 Re-examining goals

The critical factor for establishing a successful workload management environment is to fully understand the goals of the system. Now is a good time to take a step back and re-examine why it was that you purchased QP and why you configured it the way you did.

Here are some common goals of our QP customers:

- ▶ Service level agreement objectives to maintain a certain average response times.
- ▶ Service level agreement objectives for blocks of activities (batch loading of data, for example) to complete by a certain time of day.
- ▶ Prevent rogue queries from monopolizing system resources and slowing down other database activities
- ▶ Capture database activity information for charge back accounting.
- ▶ Simply maximize the throughput of data requests on the system.

13.3.3 Considerations when migrating from a QP environment

If you remember the discussion at the beginning of this book, an effective workload management is based on the following four phases to be well understood and efficiently tuned:

- ▶ Understanding of business goals.
- ▶ Ability to identify the work that maps to those goals.
- ▶ Robust management options to execute the workload in order to meet the goal metrics.
- ▶ Ample monitoring options to ensure that you are, indeed, meeting the business goals.

Let us assume that the business goals for the workload are understood. The following sections run through some of the Query Patroller and Governor configuration options and some points to ponder when considering how they may map to WLM configuration.

Identification

Query Patroller identifies and classifies database activities in two ways. The first is based on a range of estimated query cost (for example, group work by small, medium, or large sized queries) and the second is based on the user ID or group ID that submitted the query. There is also an implicit classification of work with QP simply because QP can only intercept DML (that is, all DDL, utilities, and so on will never appear on the QP radar).

DB2 WLM identifies and classifies work in many more ways. They can be grouped by the following connection attributes:

- ▶ Application name
- ▶ System user
- ▶ Session user
- ▶ A group of session users
- ▶ A role of session users
- ▶ Client user ID
- ▶ Client application name
- ▶ Client workstation name
- ▶ Client accounting string

These activities can be further classified based on the type of activities such as:

- ▶ DML
- ▶ DDL
- ▶ LOAD utility
- ▶ READ statements
- ▶ WRITE statements
- ▶ CALL (for stored procedures)

Rather than simply creating a workload based on the SESSION_USER ID or group of SESSION_USER IDs, consider widening the scope of how you may want to manage the work.

Migration action: Take advantage of the additional identification options at your disposal in DB2 WLM.

Management

In the section, we discuss the major QP management functions and their equivalents in DB2 WLM and provide migration tips.

Concurrency threshold

Query Patroller workload management relies primarily on concurrency thresholds to limit the number of queries that can run at a given time for a given classification. The premise is that queuing some of the activities on a data server that is running over capacity will prevent resource contention that can result in costly paging and thrashing. Indeed, there have been some rather dramatic improvements in workload throughput for some customers.

The actual concurrency threshold, however, can be quite difficult to come up with. How do you really know when you start to hit resource contention? Is it when Joe submits more than five queries? Does it depend on the time of day? Or volume of work?

The process of determining the threshold value often ends up being subjective and the result is a generalization of the management of the workload across users and has to be lenient enough so as not to impose on productivity throughout the day when there are varying volumes of database activity. Even so, there is still a fair amount of trial and error to come up with that configuration.

Concurrency thresholds can be useful, though, and they are available through DB2 WLM. But instead of jumping to concurrency thresholds immediately, consider using the resource controls available for a service class. Using these, you can give database activities relative priorities for CPU and prefetching (I/O). Using AIX WLM, you can even configure a more granular level of CPU resource control to make available a certain percentage of the processing power for the service class.

Migration action: Don't use concurrency thresholds as the first option for workload management. Instead, look at controlling the CPU and prefetch priority options on service classes.

Query classes

Query classes are another form of concurrency control, and one of the most powerful management options inside of QP. The concept behind query classes is to allow the query requests to be grouped by estimated cost of the query. There is often three groups: very short transaction queries, the very large report generation queries, and then some general maintenance or ad-hoc queries that fall somewhere in the middle. Each group, or query class, can be configured to limit the maximum number of queries that can run inside that group at a time.

QP is most effective when you set a low maximum concurrency level for the class of large queries. This prevents those big, resource intensive queries from consuming a disproportionate amount of the resources that would result in slower, unpredictable response times for the transaction query class.

The same issues of actually trying to determine a proper configuration for query classes as mentioned above except now, not only do you have to determine a suitable maximum concurrency, you also have to figure out what defines a small, medium and large query. Remember, QP relies heavily on cost estimates from the optimizer to define the range of cost for each query class. That estimate is very sensitive to statistics on the server and has a reputation of occasionally being inaccurate (which could result in a query being assigned to the wrong query class and hearing complaints about a normally short running query taking much longer because it was assigned to the wrong class).

It is possible to configure DB2 WLM to have the equivalent to query classes. You would use work classes and work action sets to identify the DML work and then set up thresholds on the work action set based on the range of query cost.

But this is the prime example of stepping back and re-examining the business goals. Typically, we find that query classes are used to maximize throughput by limiting the longer running queries. When you think about it though, these different types of queries rarely come from the same application. Short queries could be mass transactions, OLTP, even ETL activity. The long running queries might be heavy report generation applications. Why not create a workload for the short transactions and another for the heavy reports and map the work to separate service classes. Then it is possible to control the resources for each service class where you can keep the large queries throttled back by limiting CPU and/or I/O. Concurrency thresholds are still available on a service class, so you could still say you want to limit the number of activities for an application to 5.

Migration action: Initially, don't try to implement a query class-like environment in DB2 WLM. Look to pull database activities out into service classes based on the source attributes and control resource consumption (and possibly concurrency) for the service class.

Cost threshold

Cost thresholds are set in Query Patroller to identify queries that are either estimated to be quite small or very large.

There is a minimum cost to manage setting in QP to exclude the set of short queries from being managed. This is required because of the potentially huge volume of short queries all communicating with the query controller plus the fact that all those queries get written to a QP control table. DB2 WLM does not have these issues; the WLM logic is built into the engine (there is no communication) and the queries are not required to be written to a table in order to be managed.

The QP maximum cost threshold, however, is used to identify those queries that are estimated to be so big that you don't want to even start executing the SQL. A

typical example would be when a user mistakenly issues a query containing a Cartesian join. QP will put these queries in a held state, does not start executing the query, and returns an error to the submitting application.

It is great to be able to catch those types of queries in a WLM environment as well. A work class set could be created to identify these queries:

```
CREATE WORK CLASS SET "queries"
(WORK CLASS "very big query"
WORK TYPE DML
FOR TIMERONCOST FROM 10000000 TO UNBOUNDED)
```

There are a few actions that could be taken against these very big queries. The QP approach was to simply stop the query from running. But, like query classes, the estimated timeron cost is an estimate and has the potential to be wrong or, perhaps sometimes it really is necessary to run those big queries. In these cases, stopping the query may be too harsh an action. Collecting activity details in the event monitor might be a better solution so you analyze the potential problem later. To do this, you would create the corresponding work action set:

```
CREATE WORK ACTION SET "query handler"
FOR SERVICE CLASS "North American region"
USING WORK CLASS SET "queries"
(WORK ACTION "collect details"
ON WORK CLASS "very big query"
COLLECT ACTIVITY DATA WITH DETAILS AND VALUES)
```

If you really do want to stop those very big queries from ever running, you could create a work action set something like:

```
CREATE WORK ACTION SET "query handler"
FOR SERVICE CLASS "North American region"
USING WORK CLASS SET "queries"
(WORK ACTION "do not run"
ON WORK CLASS "very big query"
PREVENT EXECUTION)
```

Migration action: Don't be concerned about trying to implement a minimum cost to manage setting in WLM. Identifying large queries is still very useful, but consider the option of simply collecting detailed information about the very large queries before taking a harsher action.

Bypass options

A number of bypass options were introduced into Query Patroller in order to minimize the overhead of the Query Patroller tool itself. Because each managed query requires communication with the Query Patroller server and an update to

the QP control tables, it is best to set up QP to concentrate on the subset of queries that have the most dramatic affect on the performance of the data server as a whole.

Typically, queries lower than a set minimum estimated cost bypass QP along with some applications that have a fixed and known workload (e.g. batch ETL scripts).

Queries and application can bypass QP by either setting the minimum cost to manage in the submitter profile, the 'Applications to bypass' settings in the QP system properties, or by setting the <MIN_COST> or <APP_LIST> registry variables on the data server itself.

DB2 WLM is integrated into the DB2 engine and has no separate server application (like the QP server) and does not (by default) write individual records of activity details to a table so WLM does not have the overhead issues that QP has and, therefore, no option to bypass WLM is provided (not to mention that it is a fundamental architecture point to have all database activities run in a service class).

Migration action: None. There is no need and, therefore, no mechanism to allow database activities to bypass WLM.

Managing with connection pooling

Connection pooling is a very common practice found in many vendor products the access data from DB2. Basically, the tool maintains a small number of open connections to a data server, typically in a middle tier in a 3-tier environment. Users access the middle tier application and the vendor tool with a client user ID that is authenticated at the middle tier. The vendor application then knows it is OK to go get data with one of its open connections to the data server.

The problem is that QP (and DB2, for that matter) is only aware of the session user ID that connected to the database, which is not the client user ID. This makes it impossible to explicitly manage query workloads based on the client submitter.

For example, user JOE is using a vendor application to build a report. JOE connects to reportapp.exe with a user ID 'JOE' and the vendor application determines JOE is ok to run the report but it uses a DB2 connection from its connection pool that connected with the user ID 'VENDORID' to avoid having to establish and maintain a new connection for the 'JOE' user ID. QP considers the submitter of the report query to be 'VENDORID' because the user 'JOE' is never flowed to the data server. Therefore, there is no way to control Joe's query activity - not without control every other user using the connection from the connection pool, that is.

DB2 WLM, on the other hand has the client attributes available for proper identification and management of queries. The application at the middle tier could make an `sqlseti` call to set one of the client attributes before it issues the SQL.

In the example above, even though the connection is established using the 'VENDORID' session ID, the vendor application can call the `sqlseti` before the query is run to provide data for the report and set the `CURRENT CLIENT_USERID` to 'JOE' on the connection. JOE's database activity could then be controlled by limiting his resource consumption or setting thresholds on the requests.

Once a client attribute is set on the connection, it can then be used to identify database activities that are coming from the middle tier based on the actual client attributes.

Migration action: Use the `sqlseti` API to set client attributes to handle connection pooling in a 3-tier application environment.

13.3.4 Considerations when migrating from a DB2 Governor environment

The DB2 Governor has its configuration stored in a control file. There are a few parameters in particular that would be of interest when you start migrating to DB2 WLM.

The `AUTHID` and `APPLNAME` parameters are set to identify the session authorization ID and application name that the Governor should be watching. These parameters could be a good indication that there are database activities from these sources that need some attention. To create a workload based on these sources, `AUTHID` would map directly to the `SESSION_USER` workload parameter and `APPLNAME` maps to `APPLNAME` workload parameter.

There are a couple of resource events to look at as well.

The `rowssel` event is used to indicate that once a certain number of data rows are returned to the application some action should be taken. There is a DB2 WLM threshold for maximum rows returned as well.

The `idle` event is used to indicate a connection has remained idle for too long a period of time. There is a DB2 WLM threshold for maximum connection idle time as well.

If you create thresholds for the service classes created to map to the same workload the Governor was watching, consider all the threshold actions you have at your disposal. Typically the Governor forces the application when a resource threshold is exceeded. In DB2 WLM you can take a much gentler action by stopping execution of a particular activity, but you also have the option of letting the threshold continue to execute and used the information logged in the threshold violation event monitor to further investigate the problem.

Migration action: In the Governor control file, use the AUTHID and APPLNAME parameters to identify work that could map to a DB2 WLM workload and mapped to a service class. Look at the rowssel and idle resource events to possibly create corresponding WLM thresholds. However, consider the option of logging threshold violations rather than stopping execution.

13.3.5 Historical information in QP control tables

One advantage existing Query Patroller customers will have over customers new to WLM is the fact that they already have a (potentially very large) set of query activity. This is a great start towards understanding the existing workload on the data server.

The TRACK_QUERY_INFO control table in QP stores a large volume of information about all the queries that have been managed by Query Patroller. There are a number of columns in this table that map directly to connection attributes used to define a workload.

Table 13-2 displays these columns along with some suggestions of what to consider when identifying database activities through workloads.

Table 13-2 TRACK_QUERY_INFO columns that map to workload connection attributes

TRACK_QUERY_INFO Column	WLM Migration guidance
USER_ID	This column would map directly to the SESSION_USER connection attribute on the CREATE WORKLOAD statement. Look for logical patterns.

TRACK_QUERY_INFO Column	WLM Migration guidance
APPLICATION	<p>This column would map directly to the APPLNAME connection attribute on the CREATE WORKLOAD statement. It is quite likely to find that some applications are associated with a certain type of query. For example, ETL applications will usually have very short execution times (EXECUTION_TIME_SECONDS and EXECUTION_TIME_MILLISECONDS columns in the TRACK_QUERY_INFO table).</p>
CLIENT_USER_ID	<p>This column would map directly to the CURRENT_CLIENT_USERID connection attribute on the CREATE WORKLOAD statement. This column will only contain information if the submitting application invoked the sqleseti API to set this attribute. If this is the case, then this is very useful information to assign client database activities to the proper service class, especially in a multi-tier application using connection pooling.</p>
CLIENT_ACCOUNT_ID	<p>This column would map directly to the CURRENT_CLIENT_ACCTNG connection attribute on the CREATE WORKLOAD statement. This column will only contain information if the submitting application invoked the sqleseti API to set this attribute. If this is the case, this could be useful to assign database activities to a service class based on account information (or some other custom criteria depending on how this attribute is being used in the environment).</p>

TRACK_QUERY_INFO Column	WLM Migration guidance
CLIENT_APPLICATION	This column would map directly to the CURRENT_CLIENT_APPLNAME connection attribute on the CREATE WORKLOAD statement. This column will only contain information if the submitting application invoked the sqleseti API to set this attribute. If this is the case, this could be useful to assign database activities to a service class based on a custom client application name (sometimes a client process can have multiple invocations with different purposes).
CLIENT_WORKSTATION	This column would map directly to the CURRENT_CLIENT_WRKSTNNAME connection attribute on the CREATE WORKLOAD statement. This column will only contain information if the submitting application invoked the sqleseti API to set this attribute. If this is the case, this could be useful to assign database activities to a service class based on the actual workstations they are submitted from (say, when a shared terminal is used to access data).

Ideally, you would be able to create new workloads to isolate the database activities based on the analysis of the connection attribute information. Note that once those workload occurrences are assigned to a service class, Query Patroller will no longer be aware of those queries.

Figure 13-3 shows what the workload management environment would start to be shaped once you start isolating activities based on the QP history.

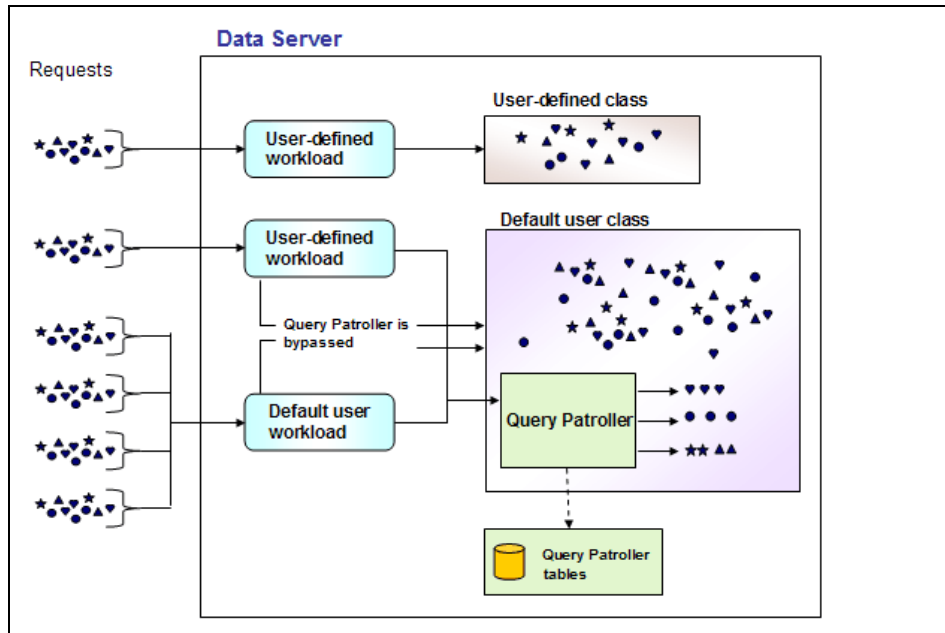


Figure 13-3 Establishing user defined service classes with QP data

There are a few other columns in the TRCK_QUERY_INFO control table that are of interest as well. These columns provide information on the type and distribution of the database requests. See Table 13-3.

Table 13-3 TRACK_QUERY_INFO columns that map to other WLM function

TRACK_QUERY_INFO Column	WLM Migration guidance
TYPE	QP intercepts all types of DML. This column will indicate if the query was a read (SELECT) or a write (INSERT, UPDATE, or DELETE). If there are patterns (for example, reads are typically very short, the rest may be long) then you could consider creating a work class or work action set to map a particular type (read or write) to a service subclass so they can be managed separately.

TRACK_QUERY_INFO Column	WLM Migration guidance
EXECUTION_TIME_SECONDS, EXECUTION_TIME_MILLISECONDS	These columns will provide the actual time a query spent inside the DB2 engine. This can be useful to try and determine the overall distribution of the query workload. Analysis can be performed on that distribution to isolate those extremely long running queries that could be disruptive to the system. This could be an indication that a threshold should be defined in order to take some action (e.g. capture details, stop execution).
ESTIMATED_COST	This column provides the estimated cost for each managed query. Like the execution time, this data can be used to analyze the distribution of expected response time of queries. An additional use for this column is to identify queries that may appear to be of a very high cost and you could consider defining a work action set to isolate those requests further and possibly even prevent them from executing. You can even look at the SQL text in these cases to look for patterns - perhaps they all include Cartesian joins and should not be allowed to run.

Related publications

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

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 391. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Leveraging DB2 Data Warehouse Edition for Business Intelligence*, SG24-7274
- ▶ *AIX 5L Workload Manager (WLM)*, SG24-5977

Other publications

These publications are also relevant as further information sources:

IBM - DB2 9

- ▶ *What's New*, SC10-4253
- ▶ *Administration Guide: Implementation*, SC10-4221
- ▶ *Administration Guide: Planning*, SC10-4223
- ▶ *Administrative API Reference*, SC10-4231
- ▶ *Administrative SQL Routines and Views*, SC10-4293
- ▶ *Administration Guide for Federated Systems*, SC19-1020
- ▶ *Call Level Interface Guide and Reference, Volume 1*, SC10-4224
- ▶ *Call Level Interface Guide and Reference, Volume 2*, SC10-4225
- ▶ *Command Reference*, SC10-4226
- ▶ *Data Movement Utilities Guide and Reference*, SC10-4227
- ▶ *Data Recovery and High Availability Guide and Reference*, SC10-4228
- ▶ *Developing ADO.NET and OLE DB Applications*, SC10-4230
- ▶ *Developing Embedded SQL Applications*, SC10-4232

- ▶ *Developing Java Applications*, SC10-4233
- ▶ *Developing Perl and PHP Applications*, SC10-4234
- ▶ *Developing SQL and External Routines*, SC10-4373
- ▶ *Getting Started with Database Application Development*, SC10-4252
- ▶ *Getting started with DB2 installation and administration on Linux and Windows*, GC10-4247
- ▶ *Message Reference Volume 1*, SC10-4238
- ▶ *Message Reference Volume 2*, SC10-4239
- ▶ *Migration Guide*, GC10-4237
- ▶ *Performance Guide*, SC10-4222
- ▶ *Query Patroller Administration and User's Guide*, GC10-4241
- ▶ *Quick Beginnings for DB2 Clients*, GC10-4242
- ▶ *Quick Beginnings for DB2 Servers*, GC10-4246
- ▶ *Spatial Extender and Geodetic Data Management Feature User's Guide and Reference*, SC18-9749
- ▶ *SQL Guide*, SC10-4248
- ▶ *SQL Reference, Volume 1*, SC10-4249
- ▶ *SQL Reference, Volume 2*, SC10-4250
- ▶ *System Monitor Guide and Reference*, SC10-4251
- ▶ *Troubleshooting Guide*, GC10-4240
- ▶ *Visual Explain Tutorial*, SC10-4319
- ▶ *XML Extender Administration and Programming*, SC18-9750
- ▶ *XML Guide*, SC10-4254
- ▶ *XQuery Reference*, SC18-9796
- ▶ *DB2 Connect User's Guide*, SC10-4229
- ▶ *DB2 9 PureXML Guide*, SG24-7315
- ▶ *Quick Beginnings for DB2 Connect Personal Edition*, GC10-4244
- ▶ *Quick Beginnings for DB2 Connect Servers*, GC10-4243

Online resources

These Web sites are also relevant as further information sources:

- ▶ DB2 Information Center
<http://publib.boulder.ibm.com/infocenter/db21uw/v9/index.jsp>
- ▶ Database and Information Management home page
<http://www.ibm.com/software/data/>
- ▶ DB2 Universal Database home page
<http://www.ibm.com/software/data/db2/udb/>
- ▶ DB2 Technical Support
<http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/index.d2w/report>
- ▶ DB2 online library
<http://www.ibm.com/db2/library>

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

ABP daemon 55
 act_total 114
 active logs 44
 activity 17
 activity estimated cost histogram 115
 activity event monitor group values
 ACTIVITY 100
 ACTIVITYSTMT 100
 ACTIVITYVALS 100
 CONTROL 100
 activity execution time histogram 114–115
 activity inter-arrival time histogram 115
 activity lifetime histogram 114
 activity monitor 161
 activity queue time histogram 115
 activity statement monitor 166
 activity threshold 164
 activity thresholds 14
 ACTIVITYSTMT_ACT_MON 166
 ad-hoc queries 174
 administration notification 360
 administration notification logs 360
 administration server 49
 administrative table function 360
 administrative view 88
 agent priority 154, 247
 aggregate thresholdan 14
 aggregated data 324
 analytic application development 235
 application name 252
 application row 322
 APPLID 163
 archived logs 44
 asynchronous background processing 55
 attribute 251
 authority 241
 authorization 359
 automatic classification 197
 automatic save 242
 availability 151
 average execution times 91

B

backup 174, 367
 bandwidth 194, 365
 BI 8
 BI perspective 240
 browse button 257
 buffer pools 44, 326
 business groups 152
 business intelligence 8
 business objective 152

C

CAP_NUMA_ATTACH 176
 CAP_PROPAGATE 176
 capability 41
 capacity planning 222
 cardinality 322
 catalog table 52
 catalog table space 367
 chuser 176
 class name 197
 clean execution 297
 collect activity data 18
 collection interval 117
 collection parameters 335
 comma separated values 218
 concurrency 286
 concurrency control 372
 concurrent versions system 242
 concurrent_act_top 114
 concurrent_connection_top 114
 concurrent_wlo_top 114
 configuration 366
 configuration process 247
 connection attributes 13
 connection idle time 170
 container for entity 244
 control rule 277
 control table 373
 coord_act_aborted_total 114
 coord_act_comp_total 136
 coord_act_completed_total 114
 coord_act_est_cost_avg 115

coord_act_exec_time_avg 114
 coord_act_interarrival_time_avg 115
 coord_act_lifetime_avg 114
 coord_act_lifetime_top 114
 coord_act_queue_time_avg 114
 coord_act_rejected_total 114
 CoordActEstCost 120
 CoordActExecTime 120
 CoordActInterArrivalTime 120
 CoordActLifeTime 120
 CoordActQueueTime 120
 coordinating partition 47
 coordinator agent 129
 coordinator partition 127
 cost thresholds 380
 cost_estimate_top 114
 count activity 18
 create a scheme by objective 246
 create a scheme yourself 246
 create limits 271
 CSV 218

D

data element 236
 data mining 235–236
 data retrieval 342
 data structure 236
 database configuration 57
 database partition 16
 database partitions 48
 database performance 57, 220
 database statistics 320
 database wide 164
 db2_install 49
 db2advis 100
 DB2CHECKCLIENTINTERVAL 172
 db2diag.log 360
 db2evmon 99
 db2look 367
 db2pd 84, 364, 366
 db2setup 49
 db2sysc 213
 deadlock detector 55
 default service class
 SYSDEFAULTMAINTENANCECLASS 53
 SYSDEFAULTSYSTEMCLASS 53
 SYSDEFAULTUSERCLASS 53
 default service classes

SYSDEFAULTMAINTENANCECLASS 9
 SYSDEFAULTSYSTEMCLASS 9
 SYSDEFAULTUSERCLASS 9
 default subclass 195
 default superclass 194, 266
 default workload 52, 251
 SYSDEFAULTADMWORKLOAD 52
 SYSDEFAULTUSERWORKLOAD 52
 delta execution 296
 deploy 297
 DESCRIBE TABLE 101
 diagnostic log 360
 disk adapter 44
 DPF 321
 drill down 344
 dropdown 324
 DSS 174
 dynamic warehousing 235

E

Eclipse platform 236
 editor 241
 enforcement scope 16
 engine dispatchable unit 55
 environment variable 156
 evaluation order 157, 261
 event monitor 55, 99, 320
 event monitor types
 activities 26
 statistics 26
 threshold 26
 executing statement 322
 execution environment 3
 execution limit 247
 explain.ddl 109

F

fenced user 49
 file systems 44
 flat mapping 203
 formatted output 118
 framework 236
 function parameter 84
 function parameters 91

G

generate code 262

global domain 17
 global enforcement scope 16
 graphical tools 42
 grid view 243

H

hardmax 197
 health monitor 55
 heath monitor 14
 hierarchy 236
 high watermark 56, 335
 histogram 28
 histogram chart 345
 historical monitoring 99
 historical performance 320
 history mode 346

I

I/O 8
 identify the work 62
 incoming connections 50
 inheritance 196
 in-memory statistics 363
 installation and ocnfiguration 321
 instance home directory 44
 Instance owner 49
 integrated development environments 236
 iostat 216

K

known_hosts file 52

L

listening port 50
 lock-wait 88
 log file reader 55
 log reader 55
 log writer 55
 logging facilities 360
 logical work grouping 10
 longest-running statements 327
 long-term performance data 348

M

manage the work 62
 management requirement 152
 maximum percentage 196

memory 42
 memory page 194
 memory set 366
 methodology 245
 millisecond 145
 monitor element 136
 monitor switch 88
 monitor the work 62
 most-read table 326
 multi-partition 321

N

navigation tree 336
 nmon 197, 216
 notification log 364

O

object definitions 367
 OLAP 9
 OLTP 9, 174
 Online Analytical Processing 9
 Online Transaction Processing 9
 open source 236
 open source community 236
 operating system 44, 193
 optimizer 372
 optimizer statistics 360
 outbound correlator 211

P

page cleaner 55
 paging space 348
 partitions 328
 PD_GET_DIAG_HIST 360
 PD_GET_DIAG_HIST return values
 APPLHANDLE 361
 APPLNAME 360
 AUTH_ID 361
 CALLEDCOMPONENT 361
 CALLEDFUNCTION 361
 CALLEDPRODUCT 361
 COMPONENT 361
 DBNAME 360
 DBPARTITIONNUM 360
 DUMPFIL 362
 EDU_ID 360
 EDUNAME 360

EVENTDESC 361
 EVENTSTACK 362
 EVENTSTATE 362
 EVENTTYPE 361
 FACILITY 360
 FIRST_EVENTQUALIFIER 361
 FIRST_EVENTQUALIFIERTYPE 361
 FULLREC 362
 FUNCTION 361
 IMPACT 360
 INSTANCENAME 360
 LEVEL 360
 MSG 361
 MSGNUM 361
 MSGTYPE 361
 OBJNAME 361
 OBJNAME_QUALIFIER 361
 OBJTYPE 361
 OSERR 361
 PID 360
 PROCESS_NAME 360
 PRODUCT 361
 RECTYPE 360
 RETCODE 361
 SECOND_EVENTQUALIFIERTYPE 361
 TID 360
 TIMESTAMP 360
 TIMEZONE 360
 PE performance database 346
 perflets 330
 performance 62
 performance counter 347
 performance counters 322, 329–330
 performance data 320
 plug-in 236
 predefined limit 14
 predictive 17
 predictive governing tool 372
 predictive threshold 17
 prefetch priority 154, 368
 prefetcher 184
 priority 153
 privilege 13, 241
 problem diagnosis 359, 366
 Properties view 241
 prune 363
 ps 197

Q

query attributes 373
 queueing boundary 17
 queueing threshold 17

R

reactive threshold 17
 real-time monitoring 99
 Recourse set 196
 Redbooks Web site 391
 Contact us xiv
 refresh rate 330
 registry value 51
 relationship 256
 repository 239
 ReqExecTime 159
 request execution time histogram 115
 request management 62
 request_exec_time_avg 115, 136
 resource allocation 4
 resource consumption 365
 resource contention 4
 resource intensive activities 4
 resource management 62
 response time 153, 161
 RESTRICT option 368
 reverse engineering 246
 rows_returned_top 114
 rsh 51
 runtime environment 236
 runtimes 236

S

SAN 44
 SAN disk 44
 scalable data warehouse 235
 scope 15
 service class 8–9, 56
 service classes 323
 service level agreement 10
 SET EVENT MONITOR STATE 102
 shared subclass 195
 shared superclass 194
 SNAP_GET_APPL_V95 158
 snapshot 93, 320, 346
 snapshot counters 330
 softmax 196
 software requirement 42

- SQL 13
 - sqleseti 383
 - ssh 51
 - stage of WLM 2
 - statement 262
 - statement cache 327
 - statistical table function 57
 - statistics collection 363
 - statistics event monitor group values
 - CONTROL 100
 - HISTOGRAMBIN 100
 - QSTATS 100
 - SCSTATS 100
 - WCSTATS 100
 - WLSTATS 100
 - storage area network 44
 - stored procedures 17
 - subagents 129
 - subclass 344
 - summary statistics 91, 132
 - superclass 238
 - system administration 14
 - system requirement 42
 - system resources 247
 - system superclass 194
 - system user 252
- T**
- table function 360
 - table spaces 326
 - tag 197
 - target database 317
 - target object 102
 - temp_tablespace_top 114
 - threshold 14–15
 - threshold boundary 17
 - threshold domain 15
 - threshold queue 26
 - threshold violation 100, 109
 - threshold violations 351
 - Thresholds 8
 - THRESHOLDVIOLATIONS_THRESH_MON 171
 - tier level 195
 - time stamp 330
 - timerons 322
 - topas 197
 - transaction log 47
 - transactional queries 374
 - tree structure 239
 - tree view 243
 - trehshold vioation event monitor group values
 - CONTROL 100
 - THRESHOLDVIOLATIONS 100
 - trend analysis 347
 - trending 222
 - troubleshooting 323
- U**
- unclassified superclass 194
 - unit of work 163
 - universal coordinated time 360
 - unmanaged superclass 194
 - UOW 163
 - USAGE privilege 368
 - UTC 360
 - utility 8
- V**
- validate 261
 - validation setting 262
 - Visual Explain 322
 - vmstat 216
 - volume group 220
- W**
- warehouse management 235
 - watermark 27
 - WLM catalog tables
 - SYSCAT.HISTOGRAMTEMPLATEBINS 30
 - SYSCAT.HISTOGRAMTEMPLATES 30
 - SYSCAT.HISTOGRAMTEMPLATEUSE 30
 - SYSCAT.SERVICECLASSES 30
 - SYSCAT.THRESHOLDS 30
 - SYSCAT.WORKACTIONS 30
 - SYSCAT.WORKACTIONSETS 30
 - SYSCAT.WORKCLASSSETS 30
 - SYSCAT.WORKLOADAUTH 30
 - SYSCAT.WORKLOADS 30
 - WLM object 236
 - WLM scheme 292
 - WLM SQL statements
 - CREATE SERVICE CLASS 31
 - CREATE THRESHOLD 34
 - CREATE WORK ACTION SET 18
 - CREATE WORK CLASS SET 36

CREATE WORKLOAD 32
 REATE WORK ACTION SET 37
 WLM stored procedures
 WLM_CANCEL_ACTIVITY 29, 95
 WLM_CAPTURE_ACTIVITY_IN_PROGRESS
 96
 WLM_COLLECT_STATS 29, 97
 WLM_SET_CLIENT_INFO 29
 WLM table functions
 GET_SERVICECLASS_WORKLOAD_OCCUR
 ENCES 57
 WLM_GET_ACTIVITY_DETAILS 25, 59, 87
 WLM_GET_QUEUE_STATS 25, 92
 WLM_GET_SERVICE_CLASS_AGENTS 24,
 58, 85
 WLM_GET_SERVICE_CLASS_WORKLOAD_
 OCCURRENCES 24, 84
 WLM_GET_SERVICE_SUBCLASS_STATS
 25, 56, 90
 WLM_GET_SERVICE_SUPERCLASS_STATS
 25, 56, 90
 WLM_GET_WORK_ACTION_SET_STATS 25
 WLM_GET_WORKLOAD_OCCURRENCE_AC
 TIVITIES 86
 WLM_GET_WORKLOAD_OCCURRENCE_AC
 TIVIWLM 58
 WLM_GET_WORKLOAD_STATS 25, 57, 92
 WML_GET_WORKLOAD_OCCURRENCE_AC
 TIVITIES 25
 WLM_COLLECT_INT 29, 363
 wlmassign 197
 wlmemon.ddl 26, 99
 wlmmon 366
 wlmperf 366
 wlmstat 197, 366
 wlo_completed_total 114
 work action 100
 Work action set 8
 work action set 17
 work class 8, 17
 work class set 17
 work identities 244
 work type 17, 244
 work type set 244
 work types
 ALL 18
 CALL 17
 DDL 18
 DML 17
 LOAD 18
 READ 17
 WRITE 17
 Workload 8
 workload 11, 42
 workload assignment 13
 workload capture level
 default 103
 detailed 104
 detailed with input data values 104
 workload definition 13, 15
 workload management 41
 workload management scheme 238, 242
 workload occurrence 13, 16, 84
 workload privilege 23
 workloads 323
 workspace 239

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)}>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine:fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

Draft Document for Review October 2, 2007 10:12 am

7524spine.fm 399

DB2 Workload Manager for Linux, UNIX, and Windows



(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(->Hide:)}>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine:fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

Draft Document for Review October 2, 2007 10:12 am

7524spine.fm 400



Draft Document for Review October 2, 2007 10:12 am

DB2 Workload Manager for Linux, UNIX, and Windows



Achieve business objectives effectively with DB2 Workload Manager

Use Performance Expert and Design Studio with DB2 WLM

Manage DB2 workloads proactively

DB2 Workload Manager (WLM) introduces a significant evolution in the capabilities available to database administrators for controlling and monitoring executing work within DB2. This new WLM technology is directly incorporated into the DB2 engine infrastructure to allow handling the higher volumes with minimal overhead. It is also enabled for tighter integration with external workload management products such as provided by AIX WLM.

This book describes DB2 WLM architecture, components, and the WLM specific SQL statements. We discuss installation, WLM methodology for customizing the DB2 WLM environment, new workload monitoring table functions, event monitors, and stored procedures. We provide examples and scenarios of using DB2 WLM to manage database activities in an OLTP, DSS, and mixed database systems. Through the use of examples, you will learn about these advanced workload management capabilities and see how they can be used to explicitly allocate CPU priority, detect and prevent “run away” queries, and to closely monitor database activity in a number of different ways.

We also discuss using Data Warehouse Edition Design Studio and DB2 Performance Expert with DB2 WLM. Finally, we give the primary differences between Workload Manager and Query Patroller as well as how they interact in DB2 9.5.

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

SG24-7524-00

ISBN