Information Management software

IBM

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
Tomoko Ichikawa
S Sadish Kumar
Marcia Miskimen
H T Morgan
Larry Pay
Tapio Väättänen

Redbooks

ibm.com/redbooks

International Technical Support Organization

DB2 Workload Manager for Linux, UNIX, and Windows

May 2008

SG24-7524-00

Note: Before using this information and the product it supports, read the information in "Notices" on page vii.

First Edition (May 2008)

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

Contents

Notices

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 higher volumes with minimal overhead. It is also enabled for tighter integration with external workload management products, such as those that are provided by AIX® WLM.

This IBM® Redbooks® publication discusses the features and functions of DB2 Workload Manager for Linux®, UNIX®, and Windows®. It describes DB2 WLM architecture, components, and the WLM-specific SQL statements. It demonstrates installation, WLM methodology for customizing the DB2 WLM environment, new workload monitoring table functions, event monitors, and stored procedures. It also provides examples and scenarios using DB2 WLM to manage database activities in DSS and OLTP 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 “runaway” queries, and to closely monitor database activity in a number of different ways.

The book also discusses using Data Warehouse Edition Design Studio and DB2 Performance Expert with DB2 WLM. Finally, the primary differences between Workload Manager and Query Patroller are explained, along with 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 worked for IBM for 17 years, including seven years of developing workload management solutions for DB2 LUW. In addition, Bill holds an Honours degree in Computing Science from Dalhousie University in Halifax, Nova Scotia.

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

S Sadish Kumar is a Senior DB2 Database Administrator in IBM India. He holds a Masters degree in Computer Science and has more than 12 years of IT experience. He has been with IBM for nine years. His areas of expertise include administration, capacity planning, performance monitoring and tuning, problem determination, and database recovery with DB2 ESE on AIX, Linux, and Windows platforms. As part of the DB2 World Trade Support level 2 Engine and Application Support team in Toronto for a year and a half, Sadish supporting customers on Intel® and UNIX platforms. He now works as a Senior DB2 DBA on systems and applications for leading financial companies in the USA on assignment from IBM India. He has co-authored the IBM Redbooks publication *DB2 UDB V7.1 Performance Tuning Guide*.

Marcia Miskimen is a Software Engineer at IBM Software Group's Silicon Valley Lab, where she is a specialist supporting Information Management Tools on LUW platforms. She has worked in the IT industry for more than 25 years, both with IBM and externally. Marcia has worked in application development, systems management, operations support, and in 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 publications on DB2 Performance Expert and DB2 Recovery Expert.

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 of experience in the Information Technology field, including 20 years of working on various aspects of DB2 development and support. Since joining IBM in 1998, H T has worked in Global Services and DB2 Lab Services as a Consulting I/T Specialist, and as a DB2 Premier Support Analyst. Prior to joining IBM, his expertise in DB2 included application development and design, relational technology research, project management, software development management, and technical support management.

Larry Pay is a DB2 consultant for Linux, UNIX, and Windows with North American Lab Services. He has eight years of experience with DB2 on AIX, Solaris™, and Linux. Before joining IBM in 1999, he worked as a DBA and Systems Analyst for 15 years, implementing and supporting multiple databases for several large companies in California. Larry's area of expertise is the implementation, support, and performance tuning of DB2 UDB data warehouses.

Tapio Väättänen is an IT Specialist with IBM Global Technology Services in Finland. He has more than 12 years of experience in the IT industry, and has spent almost half of his career as a UNIX Specialist, working on databases in UNIX and Linux environments. Tapio is an IBM Certified DB2 Administrator and is currently working on the Finnish DB2 team, providing consultation services and supporting DB2 customers focusing on high availability, performance tuning and disaster recovery solutions.

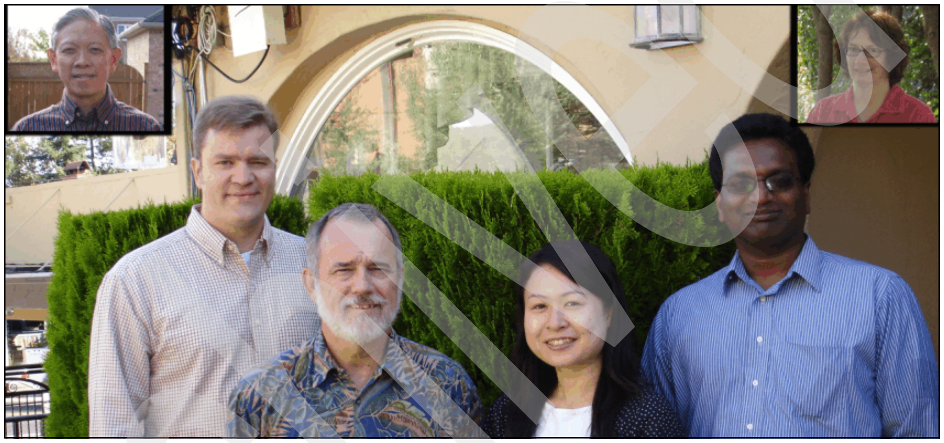


Figure 1 Left to right: Larry, Tapio, H T, Tomoko, Sadish, and Marcia. Bill Comeau (not shown) also participated in this project.

Acknowledgements

The authors express their deep gratitude for the advice and support they received from **Paul Bird** at 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

Kevin Beck
Dinkar Rao
Elizabeth Benavidez
IBM Software Group, USA

Tetsuya Shirai
IBM Systems Engineering, Japan

Anne Lesell
IBM Global Business Services, Finland

Nela Krawez
Ute Baumbach
Torsten Steinbach
IBM Software Group, Germany

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

1

Introduction

In today's world, data is being collected faster than at any other period in history. At the same time many businesses are trying to maximize their cost efficiencies by using their computing 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 trend leads to increased potential for periodic chaos in any data server environment. Thus, methods are needed for returning 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 Manager (WLM), integrated into the DB2 9.5 data server 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 Version 9.5 release.

We discuss the following topics in this chapter:

- 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, it is not uncommon to find the data server performing unexpectedly because of all the database activity. In order to keep control of the data server work, a comprehensive approach to workload management is critical.

Stages of a workload management environment

An effective workload management environment can be broken down into four basic stages, as explained here.

Define the business goals

First you need to develop 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 status meeting.

Identify the activities to be managed

After the goals are established, you need to identify the activities that you will manage. The wider the range of options for identification, the more likely it is that 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).

Manage the activities to meet the business goals

The third stage of workload management is to manage the activities previously identified in order to meet your 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 (which are a type of database activity), 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.

Monitor the activities on the data server

The final stage of workload management is the ability to monitor the state of activities on the data server. If you have taken the time to establish business goals for the environment, then it is important to have the mechanisms in place to determine whether you are meeting those goals, as well as monitoring options to identify and resolve problem areas—or even just obtain a clear picture of which activities are running. For example, if you have response time goals, then you need to be able to determine what the average response time is so you can take action if those goals are not being met.

Monitoring information is also very useful in determining if there are modifications required to the identification or management stages. It is a good idea as well 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.

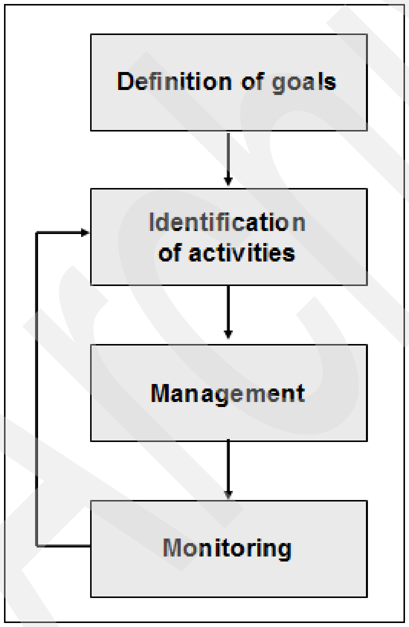


Figure 1-1 Stages of workload management

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, like a “sandbox” for the requests to “play” in.

With the execution environment concept in place, and after the business goals have been defined, the remaining stages of workload management fall into place:

- Identification

  In the identification stage, you focus on assigning database activities to an execution environment where the activities map to the activities that you built the business goals around.

- Management

  In this stage, the focus is on the tactical controls that 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 DB2 business problems that can be addressed through an effective workload management implementation. These problems include:

- Protecting the system from (or, runaway) rogue queries

  Large resource-intensive activities (complex queries, for example) 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 cases where too many 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 a 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.

- Providing 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.

- Enforcing Service Level Agreement (SLA) objectives

  Service Level Agreements often introduce explicit response time goals for database activities, with few 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 (that is, the 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 workload management concepts and architecture, how to get started using WLM with samples and best practices, as well as tool integration and problem determination suggestions.

2

Workload Manager architecture and features

In this chapter we describe the DB2 9.5 Workload Manager (WLM) architecture and components, and explain 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 is also examined.

Prior to DB2 V9.5, workload management meant the use of DB2 Query Patroller in conjunction with the DB2 Governor. In DB2 9.5, workload management means the use of DB2 Workload Manager by itself, in conjunction with the DB2 Governor, or coexisting with the DB2 Query Patroller and the 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 fewer resources, in an accelerated tomfooleries, 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, as well as 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 work in the system to slow down. Some companies would acquire more hardware resources to address the problem. Other companies would terminate the resource-intensive queries outright. Still others might choose to tune system performance to recover production capacity, or 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 new set of features was architected and introduced into the DB2 9.5 engine in the form of DB2 Workload Manager (WLM).

DB2 WLM is a powerful solution that addresses these multiple issues. It allows the user to treat different workloads (applications, users, and so on) differently, and provide them with different execution environments or sandboxes to run in. The quick, flexible, and robust methodology offered by WLM can help you identify, manage, and control your workloads 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 and Work classes sets

The DB2 WLM architecture revolves around defining execution environments for activities, and assigning resources to those execution environments.The DB2 9.5 implementation starts with CPU and I/O resource management for DB2 workloads, including asking how can resource sharing be done effectively, as well as 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

The *service class* is the primary point for resource assignment to all database requests. 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. Because 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: This default user service superclass tracks all the user-defined workloads that are assigned to the default user workload.
- SYSDEFAULTSYSTEMCLASS: The default system service superclass tracks internal DB2 connections and threads that perform system-level tasks.

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.

- SYSDEFAULTMAINTENANCECLASS: The default maintenance server superclass tracks the internal DB2 connections that perform database maintenance and administration tasks. DB2 internal maintenance requests such as DB2 asynchronous background processing agents and DB2 Heath monitor initiated utilities are directed to the SYSDEFAULTMAINTENANCECLASS service class.

Each of these superclasses has a default subclass SYSDEFAULTSUBCLASS. Superclass is really a logical entity that allows for shared context for any subclasses within it. All work actually executes in a subclass. Anything assigned to a superclass executes in the default subclass created automatically for that superclass.

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.

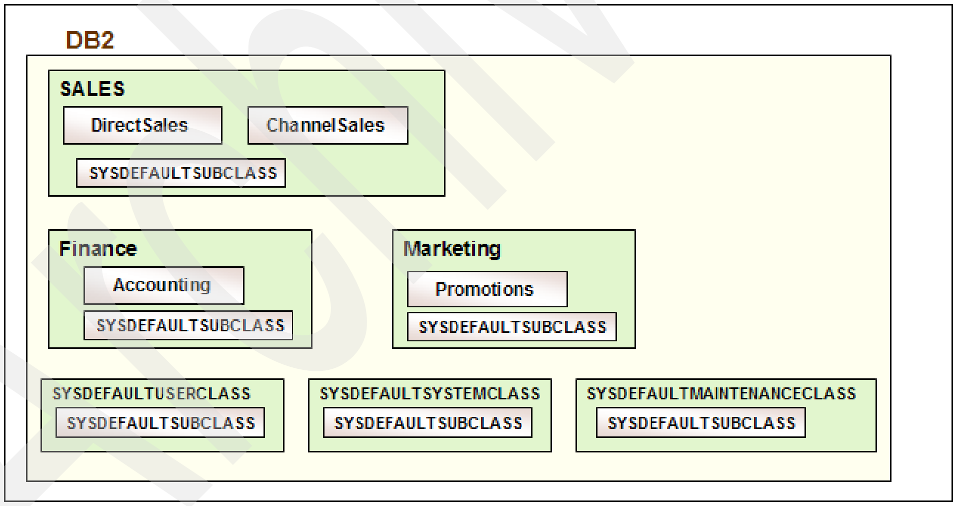


Figure 2-1 WLM service classes

Because 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 (SLAs)
- 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 groupings
- 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 displays the SQL statements to create service superclasses and service subclasses that are 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 can be tightly 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 6, “AIX Workload Manager considerations” on page 143.

2.1.2 DB2 workloads

A *DB2 workload* is an object that 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 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 cases, there are data warehouse applications that 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 - The name of the application running at the client, as known to the database server. It is specified as APPLNAME in the workload definition statement.
- System authorization ID - Authorization ID of the user that connected to the database, as set in the SYSTEM_USER special register. It is specified as SYSTEM_USER in the workload definition statement.
- Session authorization ID - Authorization ID that is used for the current session of the application, as set in the SESSION_USER special register. It is specified as SESSION_USER in the workload definition statement.
- Group of session authorization ID - Secondary authorization ID of type group of the session authorization ID. It is specified as SESSION_USER_GROUP in the workload definition statement.
- Role of session authorization ID - Secondary authorization ID of type role of the session authorization ID. It is specified as SESSION_USER_ROLE in the workload definition statement.
- Client user ID - The client user ID from the client information specified for the connection, as set in the CURRENT CLIENT_USERID (or CLIENT USERID)

special register. It is specified as CURRENT CLIENT_USERID in the workload definition statement.

- Client application name - The application name from the client information specified for the connection, as set in the CURRENT CLIENT_APPLNAME (or CLIENT APPLNAME) special register. It is specified as CURRENT CLIENT_APPLNAME in the workload definition statement.
- Client workstation name - The workstation name from the client information specified for the connection, as set in the CURRENT CLIENT_WRKSTNNAME (or CLIENT WRKSTNNAME) special register. It is specified as CURRENT CLIENT_WORKSTNNAME in the workload definition statement.
- Client accounting string - The accounting string from the client information specified for the connection, as set in the CURRENT CLIENT_ACCTNG (or CLIENT ACCTNG) special register. It is specified as CURRENT CLIENT_ACCTNG in the workload definition statement.

These connection attributes determine how a connection is mapped to a workload definition. The workload determines which service class user requests from that connection go to. Figure 2-2 shows two user-defined workloads CAMPAIGN and NEWCAMPAIGN, as well as the connection attributes used.

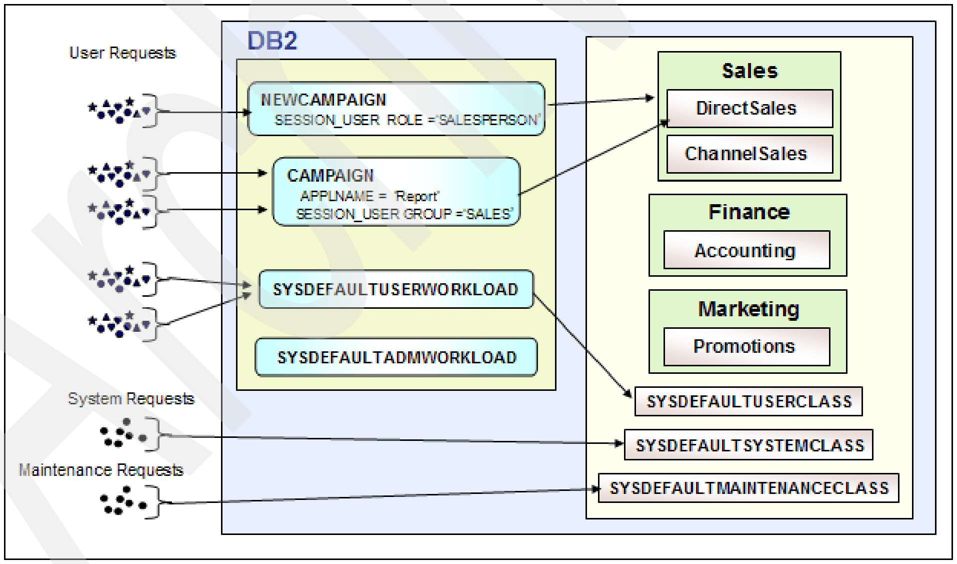


Figure 2-2 DB2 workload

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 connection is assigned to the default workload, SYSDEFAULTUSERWORKLOAD. Without re-mapping, these connections are directed to the SYSDEFAULTUSERCLASS service class.

The default system administration workload SYSDEFAULTADMWORKLOAD is a special DB2 workload that is primarily used by database administrators to perform their work or take corrective action. All workloads except the SYSDEFAULTADMWORKLOAD can have thresholds applied to them.

By default, the sequence of how the workloads are defined determines the workload evaluation order. The new workload is positioned after the other defined workloads in the evaluation order, but before the SYSDEFAULTUSERWORKLOAD workload.

You can set the evaluation order by using the POSITION keyword of the CREATE WORKLOAD or ALTER WORKLOAD statement. In Example 2-2 on page 14, the workload NEWCAMPAIGN is defined with position before CAMPAIGN. If user Bob runs the application with APPLNAME “Report” and 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 a specific criteria such as consumption of a specific resource or duration of time. 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 tempspace 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 entire Sales department may want 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 setting the maximum number of queued connections to 5. If these limits are exceeded, the application will be stopped.

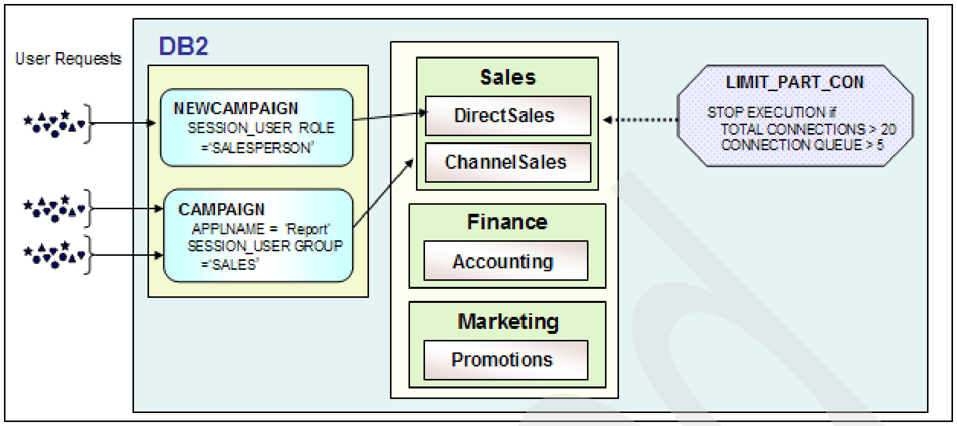


Figure 2-3 WLM threshold

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;
```

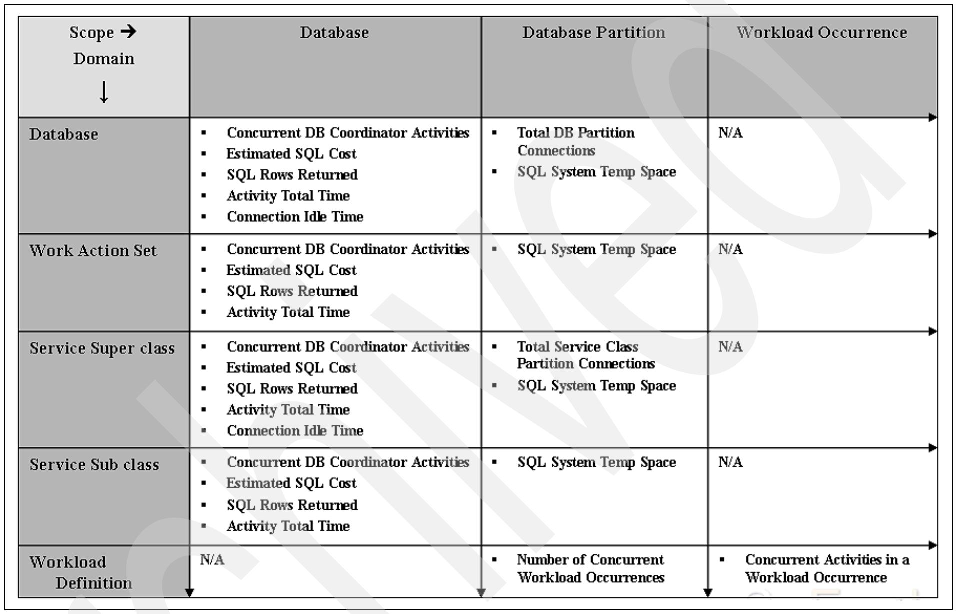
Thresholds can be enforced on activities that are part of a threshold domain, which can range from the entire database to a single workload definition. For example, if the threshold domain is a database, an enforcement scope of the threshold may be just one database partition, or all of the database 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	▪ Concurrent DB Coordinator Activities ▪ Estimated SQL Cost ▪ SQL Rows Returned ▪ Activity Total Time ▪ Connection Idle Time	▪ Total DB Partition Connections ▪ SQL System Temp Space	N/A
Work Action Set	▪ Concurrent DB Coordinator Activities ▪ Estimated SQL Cost ▪ SQL Rows Returned ▪ Activity Total Time	▪ SQL System Temp Space	N/A
Service Super class	▪ Concurrent DB Coordinator Activities ▪ Estimated SQL Cost ▪ SQL Rows Returned ▪ Activity Total Time ▪ Connection Idle Time	▪ Total Service Class Partition Connections ▪ SQL System Temp Space	N/A
Service Sub class	▪ Concurrent DB Coordinator Activities ▪ Estimated SQL Cost ▪ SQL Rows Returned ▪ Activity Total Time	▪ SQL System Temp Space	N/A
Workload Definition	N/A	▪ Number of Concurrent Workload Occurrences	▪ 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, as explained here:

- A *predictive* threshold means the threshold boundaries are checked before the tracked work is started.
- 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 may want to treat stored procedures differently from all other read and write activity against the database. In one instance, you may want to put a restriction on the number of concurrent Loads executing at any one time. You may also want Data Definition Language (DDL) put into a service class by itself. All these tasks can be accomplished by using a *work action set*.

Work action sets allow you to apply DB2 thresholds with discrimination if applied at the database level. They can be used to map activities to service subclasses with discrimination if applied at the service superclass level.

Work action sets work hand-in-hand with *work class sets*. A work class set defines the characteristics of the work of interest. In contrast, a work action set dictates what is to happen when the work of interest is detected. A work class set can be shared and used by different work action sets. Work classes can have more than one work action applied to them.

A work class has an associated work type; the supported work types are listed here:

- READ - for read-related activities such as SELECT, XQuery
- WRITE - for update related activities such as DELETE, INSERT, UPDATE
- CALL - for a CALL statement
- DML - for data manipulating activities such as SELECT, UPDATE, MERGE
- DDL - for data definition activities such as CREATE, ALTER, COMMIT
- LOAD - for the LOAD utility
- ALL - for all database activities

Work actions are grouped into work action sets. A work action can apply to either activities at the database level, or to activities at the service superclass level depending on the object that the work action set containing the work action is applied to.

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 set, and associate the work action with an existing work class within the same work class set that the work action set was associated with.
- Associate the work action set 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.

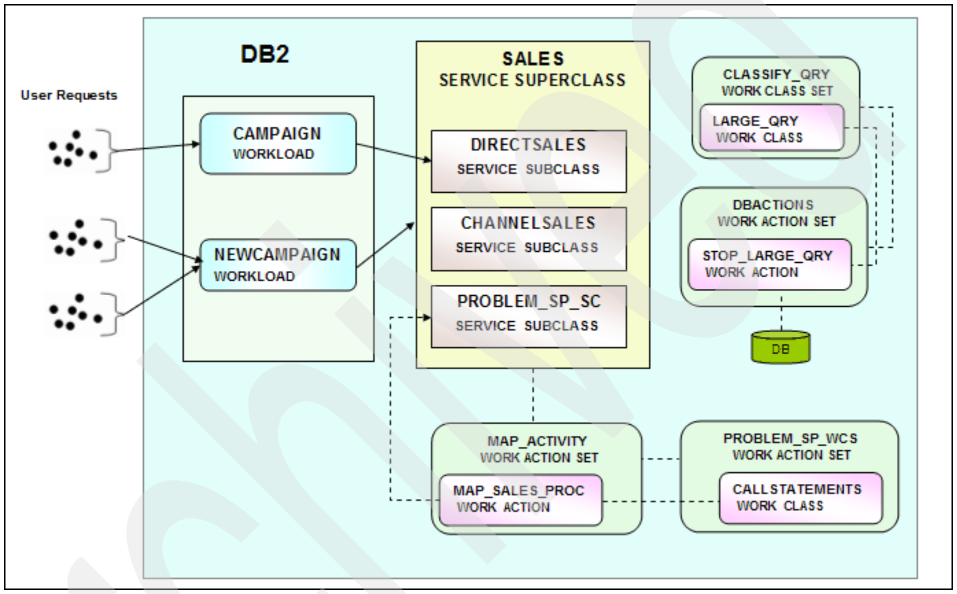
The following actions can be performed within a DB2 work action set:

- Count activity
- Prevent execution of an activity
- Collect activity data
- Map work to a different service sub class within the same superclass
- Apply a threshold to an activity that falls within the work class the work action is applied to, if the work action set is applied at the database level.
- Collect aggregate activity data for activities associated with the work class for which this work action is defined, when the work action set that the work action is in, is applied to a service superclass.

Figure 2-5 illustrates using work class sets and work action sets to manage workloads. In this example, there are two goals desired. The first goal is to stop expensive queries costing 1000001 timerons or more, and the second goal is to gather more information about stored procedures running in the SALES service superclass.

To achieve these goals, work class sets must first be created. A work class set named CLASSIFY_QRY is associated with the work class LARGE_QRY having a work type of READ. A work action set called DBACTIONS is associated with the database, and the work action STOP_LARGE_QRY is associated with the work class LARGE_QRY. When an incoming database request of work type READ with a timeron cost greater or equal to 1000001 is encountered, the execution of this database request is stopped, and data about this request is collected for analysis.

To gather more information about stored procedures, an additional service subclass called PROBLEM_SP_SC was created under service class SALES to collect activity data on the coordinator node. A work class set called PROBLEM_SP_WCS is created and associated with the work class CALLSTATEMENTS having a work type of CALL. A work action set called MAP_ACTIVITY is associated with the service super class SALES and the work class set PROBLEM_SP_WCS. When a stored procedure with a schema of SALES is executed, the work action MAP_SALES_PROC will map all activity and trigger the collection of data about the stored procedure.



DB2
User Requests
CAMPAIGN WORKLOAD
NEWCAMPAIGN WORKLOAD
SALES SERVICE SUPERCLASS
DIRECTSALES SERVICE SUBCLASS
CHANNELSALES SERVICE SUBCLASS
PROBLEM_SP_SC SERVICE SUBCLASS
CLASSIFY_QRY WORK CLASS SET
LARGE_QRY WORK CLASS
DBACTIONS WORK ACTION SET
STOP_LARGE_QRY WORK ACTION
DB
MAP_ACTIVITY WORK ACTION SET
MAP_SALES_PROC WORK ACTION
PROBLEM_SP_WCS WORK ACTION SET
CALLSTATEMENTS WORK CLASS

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 SERVICE CLASS PROBLEM_SP_SC UNDER SALES COLLECT ACTIVITY DATA ON
COORDINATOR WITH DETAILS

CREATE WORK CLASS SET CLASSIFY_QRY (WORK CLASS LARGE_QRY WORK TYPE READ FOR
TIMERONCOST FROM 1000001 To UNBOUNDED)

CREATE WORK ACTION SET DBACTIONS FOR DATABASE USING WORK CLASS SET CLASSIFY_QRY
(WORK ACTION STOP_LARGE_QRY ON WORK CLASS LARGE_QRY
WHEN ESTIMATEDSQLCOST > 1000001 COLLECT ACTIVITY DATA STOP EXECUTION)

CREATE WORK CLASS SET PROBLEM_SP_WCS (WORK CLASS CALLSTATEMENTS WORK TYPE CALL
ROUTINES IN SCHEMA "SALES")
```

```
CREATE WORK ACTION SET MAP_ACTIVITY FOR SERVICE CLASS SALES
USING WORK CLASS SET PROBLEM_SP_WCS
(WORK ACTION MAP_SALES_PROC ON WORK CLASS CALLSTATEMENTS
MAP ACTIVITY WITH NESTED TO PROBLEM_SP_SC)
```

2.2 Architecture

The architecture of the DB2 Workload Manager integrates all the workload management objects into a coherent whole in order to make it easier to identify, manage, and monitor all 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. 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 more effectively 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.

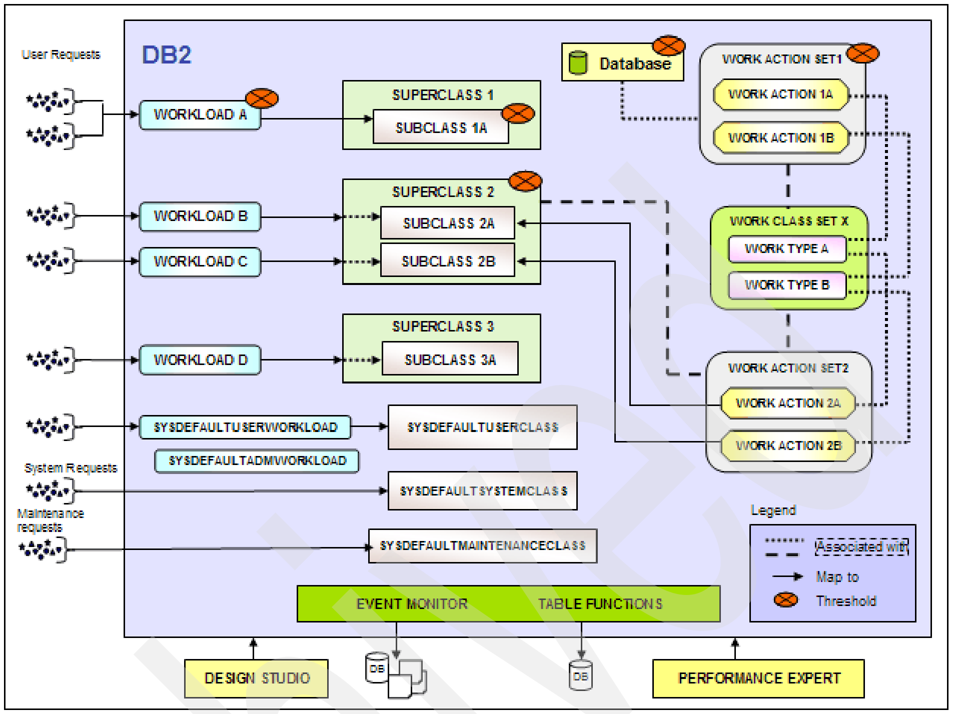


Figure 2-6 DB2 WLM architecture

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 SUPERCLASS 1 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 that 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 1 B 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. 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:

- ► DB2 service superclasses 1 and 2 are associated with AIX WLM service classes _DB2_SUPERCLASS 1 and _DB2_SUPERCLASS 2, respectively.
- ► DB2 service subclasses 1A, 2A and 2B are associated with AIX WLM subclasses _DB2_SUBCLASS 1A, _DB2_SUBCLASS 2A and _DB2_SUBCLASS_2B, respectively.
- ► DB2 service class SYSDEFAULTUSERCLASS is associated with AIX WLM service class _DB2_DEF_USER.
- ► 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 6, "AIX Workload Manager considerations" on page 143.

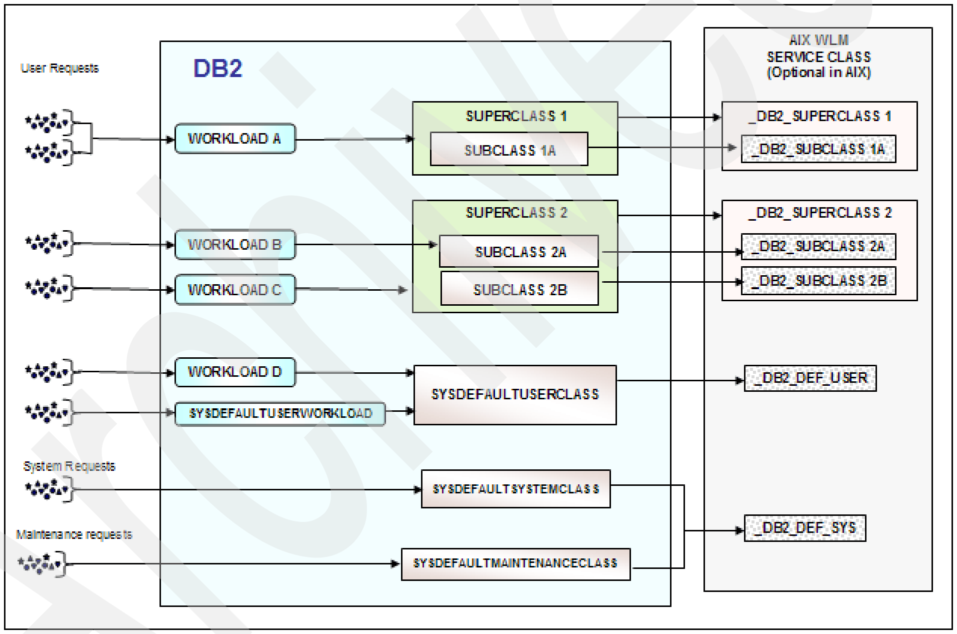


Figure 2-7 DB2 WLM integrated with AIX WLM

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 32.

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 8.1, "DB2 Warehouse Design Studio overview" on page 188.

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 WLM statistics table functions are listed here:

- 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 all 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 71 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.
- Table functions look at and gather point-in-time information.

Unlike the non-WLM event monitors, the WLM event monitors do not have event conditions that 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 timeframe. 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 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 here.

- Threshold queues:
  - Total queue assignments
  - Top queue size
  - Total queue time
- Service subclasses:
  - High watermark concurrent activity
  - Total coordinator activities completed
  - Total coordinator activities aborted
  - Total coordinator activities rejected
  - Number of active requests
- Service superclasses:
  - Top concurrent connection
- Workloads:
  - Total workload occurrences completed
  - High water mark of concurrent workload occurrences
  - High water mark of Concurrent activity
  - Total coordinator activities completed
  - Total coordinator activities aborted
  - Total Coordinator activities rejected
  - 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
- 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
- Coordinator activity inter-arrival time 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.

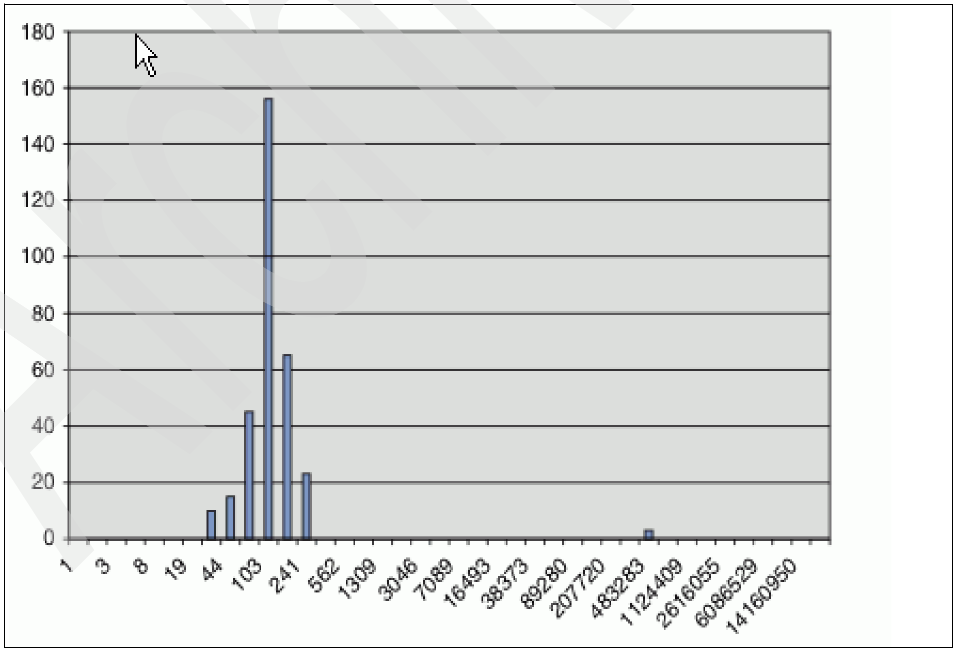


Figure 2-8 Histogram plotted to a bar chart

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 DATA BASE
- COLLECT AGGREGATE ACTIVITY DATA 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 them to the statistics event monitor.

WLM_SET_CLIENT_INFO is a stored procedure that allows you to set the values of any of the client information fields at the DB2 server by 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 it 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 current interval since the last 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/v9r5/index.jsp

- DB2 manuals:

  SQL Reference, Volume 1, SC23-5861
  SQL Reference, Volume 2, SC23-5862

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

  Use this clause to collect statistics information for service subclass only.

- COLLECT AGGREGATE REQUEST DATA clause

  Use this clause to collect aggregate request data for service class and send it to the applicable event monitor.

- 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
```

Note the following points:

- On UNIX and Linux operating systems, AGENT PRIORITY values range from -20 to 20. Negative values denote a higher relative priority.
- On Windows operating systems, AGENT PRIORITY values range from -6 to 6. Negative AGENT PRIORITY values denote a lower relative priority.

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.

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 connection attributes are case-sensitive in this statement.

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.

  If you use a customized security plug-in for authentication, you can return a different value for the current session 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 or not a request is allowed access to the database.
  - 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 a 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 have their 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

    Use this clause to enable collection of information about activities that violate the threshold. Activity information is sent to the active activities event monitor upon activity completion.
- 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 that runs in the department superclass across all database partitions.

2.5.4 DB2 Work Classes and Work Class Sets

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	► All SELECT statements (select into, values into, full select) ► SELECT statements containing a DELETE, INSERT, or UPDATE are not included ► All XQuery statements
WRITE	► All UPDATE statements (searched, positioned) ► All DELETE statements (searched, positioned) ► All INSERT statements (values, subselect) ► All MERGE statements ► All SELECT statements containing a DELETE, INSERT, or UPDATE statement
CALL	► CALL statement ► The CALL statement is only classified under the CALL and ALL work class types.
DML	All statements that are classified under the READ and WRITE work class types
DDL	► 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	► Load utility. ► The load utility is only classified under the LOAD and ALL work class types.
ALL	► 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 Actions and Work Action Sets

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 / QUEUETIME / 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)
```

2.6 DB2 WLM setup

The base function of DB2 9.5 Workload Manager is available in the core DB2 engine. The more advanced custom features are licensed under the Performance Optimization feature. There are no special considerations for the database using WLM function. Plan your DB2 environment so that it is efficient, easily recoverable, and easy to maintain. Refer to DB2 documentation for system requirements and installation procedure:

- *Getting Started with DB2 installation and administration on Linux and Windows,* GC23-5857
- *Quick Beginnings for DB2 Servers,* GC23-5864

To fully utilize DB2 WLM enhancements, plan how your applications will 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 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.

2.6.1 Lab environment

We used the following lab systems for writing this book. You may see the server names in the examples.

- AIX servers: Clyde and Bonnie
- Linux server: Puget
- Windows server: Cetus

AIX server configuration

Figure 2-9 illustrates disk configuration for the partitioned database in the AIX servers. The operating system has its own volume group, rootvg, which is on internal disks. All database-related file systems reside on the external SAN disks.

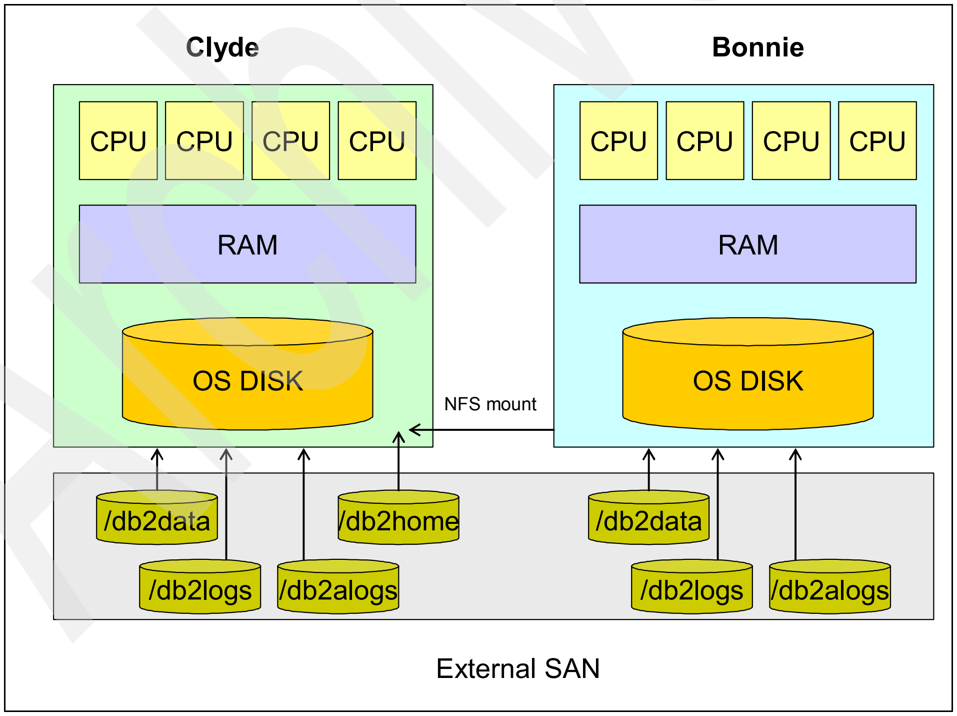


Figure 2-9 Our disk setup

Figure 2-10 illustrates the partition configuration of the WLMDB database that we created on the AIX systems. WLMDB spreads over two physical nodes, and it consists of five database partitions. Partition 0 on Clyde acts as the coordinating partition. WLMDB has six database partition groups:

- IBMCATGROUP for system catalogs on partition0, the coordinating partition.
- IBMDEFAULTGROUP spans 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 spans 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 all database partitions. It has only one table space, MAINT, for event monitor data to be used for historical analyses.

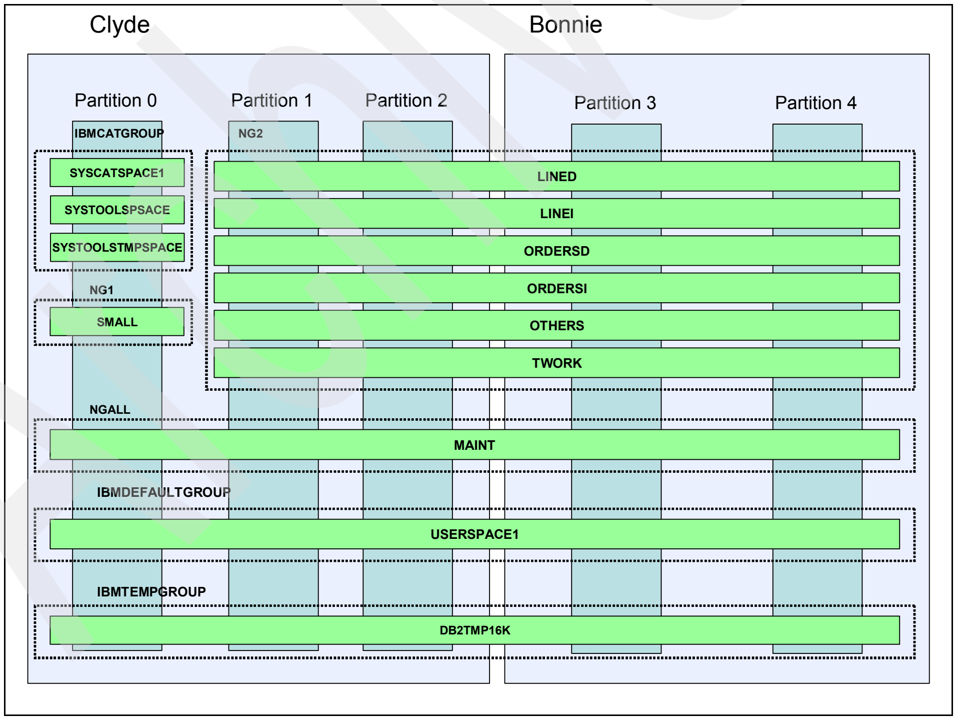


Figure 2-10 Our WLMDB partition configuration

TPC-H

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

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

Verify your installation

After DB2 is installed, you can check whether the DB2 installed has the WLM feature by creating a database and verifying if one of the new WLM catalog tables is in place. Example 2-11 shows that we created a database WLMDB and selected WLM catalog table SYSCAT.WORKLOADS. Two default workloads, `SYSDEFAULTUSERWORKLOAD` and `SYSDEFAULTADMWORKLOAD`, were created.

Example 2-11 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.
```

2.6.2 First step

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

The default DB2 WLM configuration

In DB2 9.5, DB2 creates three default service classes and two default workloads 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 2-11 illustrates the relationships between the default workloads and the service classes.

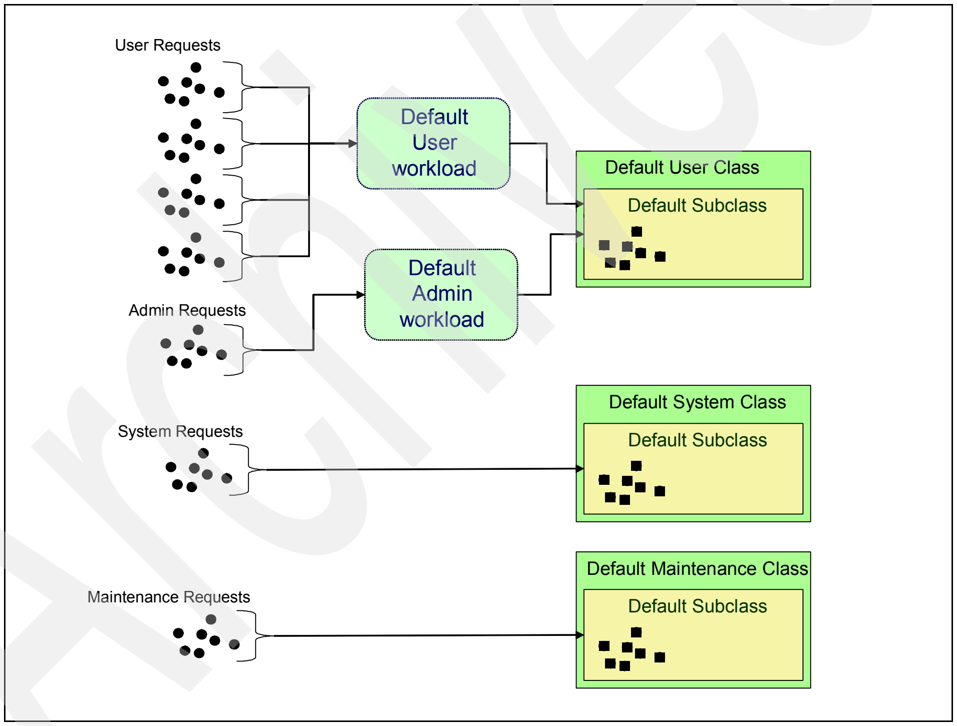


Figure 2-11 Default workloads and service classes

On a default configuration, SYSCAT.SERVICECLASSES, SYSCAT.WORKLOADS, and SYSCAT.SYSDEFAULTHISTOGRAM catalog views have content populated.

Monitoring the default WLM environment

The default WLM environment is a useful point from which to start to understand the activities taking place at the database. Even with the basic WLM environment, there is significant information available.

The monitoring functions are immediately useful because in DB2 WLM, by default, all requests run in a service class and all connections are associated with a workload. Therefore, you can immediately take advantage of the monitoring capabilities offered for workloads and service classes. 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 provide a few examples showing how to use WLM table functions to see the default WLM settings and the activities in the DB2. A detailed description of the WLM table functions is provided in Chapter 4, "Monitoring DB2 workload management information" on page 75.

- Use the service class statistics function to understand number of activities being run in the system.

  In Example 2-12, we show how to obtain the system activities information since the last statistics reset. We learned that in our test system, the highest number of concurrent activities was 4, the number of activities aborted was 3, and 24 activities completed since the last statistics reset. Such information can be very useful in helping you to examine the load on your database environment.

  Example 2-12 Summary statistics at service subclass level

```
SELECT CONCURRENT_ACT_TOP AS ACT_TOP,
       COORD_ACT_ABORTED_TOTAL AS ABORTED,
       COORD_ACT_COMPLETED_TOTAL AS COMPLETED,
       LAST_RESET
FROM
TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('SYSDEFAULTUSERCLASS','',-2));

ACT_TOP ABORTED COMPLETED LAST_RESET
------- ------- --------- --------------------------
      4       3        24 2007-11-13-15.14.41.513892

  1 record(s) selected.
```

- Use workload statistics to understand the total number of connections and the highest number of concurrent connections.

  Example 2-13 shows how to use the `WLM_GET_WORKLOAD_STATS` table function to obtain the summary statistics at the workload level. The example shows that the highest concurrent number of connections was 4 and that the total number of completed connections was 197 for workload SYSDEFAULTUSERWORKLOAD, since the last reset.

By looking at this information and the service subclasses statistics (shown in Example 2-12) over time, you can achieve a useful understanding of your database system load.

Example 2-13 Summary statistics at the workload level

```
SELECT CONCURRENT_WLO_TOP,
       COORD_ACT_COMPLETED_TOTAL
FROM TABLE(WLM_GET_WORKLOAD_STATS('SYSDEFAULTUSERWORKLOAD',-2));

CONCURRENT_WLO_TOP COORD_ACT_COMPLETED_TOTAL
------------------ -------------------------
                 4                       197

  1 record(s) selected.
```

- Use the workload occurrences function to examine the connections on the system. In particular, use this function to understand who is submitting work into system (that is, which users or which applications), and also to learn about the connection attributes for the applications that submit work to the system. This information is particularly useful if you want to isolate those applications.

  Example 2-14 shows how to list the workload occurrences on a particular service class by using the `WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES` table function.

Example 2-14 List of workload occurrences on a particular service class

```
SELECT SUBSTR(SESSION_AUTH_ID,1,15) AS SESSION_AUTH_ID,
       SUBSTR(APPLICATION_NAME,1,16) AS APPLICATION_NAME
FROM
TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('SYSDEFAULTUSERWORK
LOAD','',-2));

SESSION_AUTH_ID APPLICATION_NAME
--------------- ----------------
SAP_USER        SAP.EXE
OLTP_USER       OLTP.EXE
BATCH_USER      BATCH.EXE

  3 record(s) selected
```

- Use the workload activities table function to understand the types of activities that are run on the system. For example, you could periodically run a query that counts number of activities of a specific type. Over time, you would build up an understanding of the number of loads, the number of reads, and so on.

  Example 2-15 shows using the `WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES` table function to list queries and query types running on the system. We see that there are currently two reading activities and one writing activity. By collecting this information periodically over a time frame, you can learn how activity types and the quantity of each activity differ over time.

  Example 2-15 List queries and query types running on the system

```
SELECT APPLICATION_HANDLE AS HANDLE,
       ACTIVITY_ID AS ID,
       SUBSTR(ACTIVITY_STATE,1,9) AS STATE,
       SUBSTR(ACTIVITY_TYPE,1,8) AS TYPE
FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(CAST(NULL AS
BIGINT), -2));


HANDLE A_ID U_ID STATE     TYPE
------ ---- ---- --------- --------
   301    1    1 EXECUTING READ_DML
   420    1   10 EXECUTING WRITE_DML
   598    1    1 EXECUTING READ_DML

  3 record(s) selected.
```

- Use the workload activity details function to examine a long-running activity and understand where it came from, what statement is being run, and why it is running so long. This information can be useful later for establishing thresholds.

  In Example 2-16, we use the `WLM_GET_ACTIVITY_DETAILS` table function to find out start time, effective isolation level, rows fetched, and rows modified for application 420. The application handle number 420 is obtained from Example 2-15.

  Example 2-16 Workload activity details

```
SELECT SUBSTR(NAME, 1, 20) AS NAME,
         SUBSTR(VALUE, 1, 30) AS VALUE
  FROM TABLE(WLM_GET_ACTIVITY_DETAILS(420, 10, 1, -2))
  WHERE NAME IN ('APPLICATION_HANDLE',
                 'LOCAL_START_TIME',
                 'UOW_ID',
               'ACTIVITY_ID',
               'ACTIVITY_TYPE',
               'EFFECTIVE_ISOLATION',
               'ROWS_FETCHED',
               'ROWS_MODIFIED');

NAME                 VALUE
-------------------- ------------------------------
ACTIVITY_ID          1
ACTIVITY_TYPE        WRITE_DML
APPLICATION_HANDLE   420
LOCAL_START_TIME     2007-11-14-04.43.00.004358
UOW_ID               10
EFFECTIVE_ISOLATION  1
ROWS_FETCHED         6
ROWS_MODIFIED        21

  8 record(s) selected.
```

Customizing the WLM execution environments

Now that we have worked through the default WLM setup, it is time to begin building a WLM plan and implementing the plan. From this plan we can evolve our WLM implementation.

We discuss the following topics in this chapter:

- Steps for customizing DB2 WLM to achieve business objectives, including developing a workload identification worksheet
- Methodology for building and evolving a WLM implementation

3.1 Stages of workload management

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

As mentioned in 1.1, "Workload management" on page 2, these stages are important for managing these workloads and protecting their needed resources:

- Identify the work

  The management goals are either given or can be derived from business objectives, service delivery, or performance objectives. 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 the resources being consumed in order to meet your goals. This is where you determine what actions you want to take to try to insure that all work is completed when needed. That is, now that the workload has been identified, it now must be managed.

  Through WLM, work is prioritized and action is taken based on how the work is classified or managed. Based on how the work is managed, priorities and actions can be set. Workloads are managed by controlling the resources they may consume and their priorities for consuming them.

  Work can be managed based on the type of request and where it came from (such as ad hoc queries from the Sales department, or reports from the Accounting department). Alternatively, workloads can be managed based on a common action, function or resources consumed (such as ETL jobs loading massive amounts of data that are added to the data warehouse using the LOAD utility). These jobs may tie up tables in the database for long periods of time, so their consumption of resources may need to be controlled as to not interfere with other work needing to be complete. The actions or restrictions you want to impose on your workloads will determine how you manage them.

- Monitor the work

  Monitoring helps you to determine whether you are achieving a goal, and to identify problems that might be preventing you from achieving that goal. Using monitoring, you can capture the activity information, store it, and then analyze the data.

This is a continuous cycle that can be expressed as the cyclical process as shown in Figure 3-1.

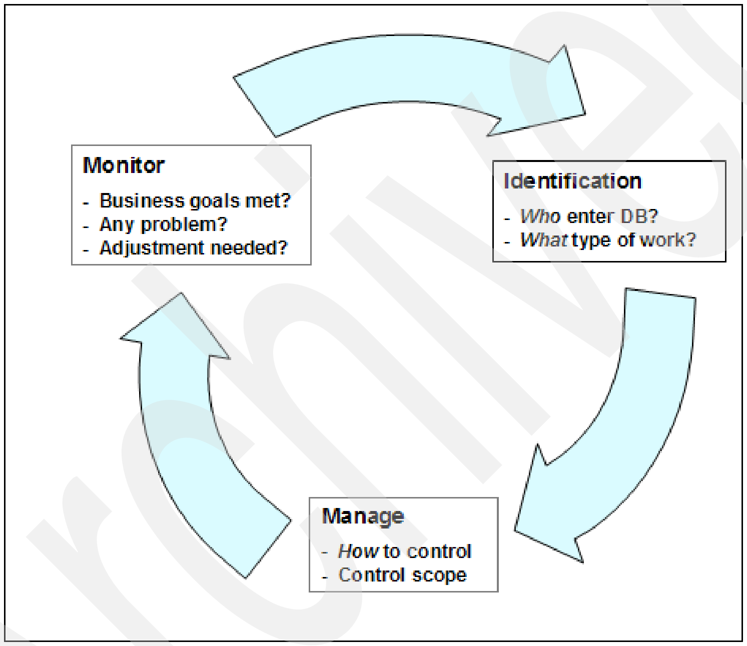


Figure 3-1 WLM methodology cycle

In the following sections, we explain these stages in greater detail.

3.2 Identify the work

The process of identifying workloads can be very complex. It is best to start by using what you already know. Much important information about the workload is often available at the very beginning of the planning process and is based on business requirements:

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

Using this information, first develop a comprehensive view of the work in the database that needs to satisfy these business requirements. Next, identify the characteristics and changes to the workload by using workload profiling.

Checklist

In the process of identifying the workload, gather as much as information about it as possible. In the management stage, this information will be used to customize the WLM to manage your workload to meet the business goal. Here we list the items to be identified:

- Tasks

  What processes do you want to identify, or what you have discovered from previous monitoring?

  Examples of processes to identify include utilities run by DBA, all the report jobs, and so on.

- Business requirements

  What is the purpose or function of a 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.

- Task identity

  How is the process identified in the system?

  WLM will identify what workload comes to the database system; it can identify a process or workload by 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? 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

    You specify what to control and the control scope by using `WORK ACTION` or `THRESHOLD`.

3.2.1 Workload identification worksheet

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

The workload example used here is a typical starting point for many data warehouse customers. We categorized 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 queries 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, and so on. They are identified by their executable name. In our example, they run dss.exe.

- Note that no service class is created for all of Production; it was merely a label for the production subcategories underneath the "Production" heading.

We began by determining what resources these workloads consumed, how often they run, and when they run. Such information can help us decide whether we need to take any action to protect an individual workload, as well as what actions we might need to take to limit the workload's impact on other workloads.

Table 3-1 is a worksheet that shows our starting point. Remember, workload management is a *continuous* cycle of identification, management, and monitoring, so this is merely a starting point of identification. After we gather more information and knowledge about our workloads, we can develop a more comprehensive plan.

This worksheet includes all the stages of workload management previously discussed. As you identify your workloads, you can then determine what requirements are needed for managing them, how they are identified in the system, and what actions are needed to mange them.

Note that this example is a simple one that illustrates the concept of documenting workload in business terms, and not strictly in technical terms. By using a worksheet similar to Table 3-1, management and users can become involved in the workload management process. They can understand and agree on the requirements and actions that are needed from a technical perspective to protect and manage the database resources.

Table 3-1 Workload identification 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

Task	Business requirements	Identification	Action
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

The headings in this worksheet are explained here:

- Task

  This indicates the workload that is being uniquely identified. (Note that "Production" is merely a label for the subtasks below and does not have any actions specifically for all of production.)

- Business Requirement

  This indicates the business requirements or rules that exist regarding this uniquely identified workload.

- Identification

  This indicates how this workload is identified in the system.

- Action

  This indicates what task will be done to manage the workload, or what will be reported about the workload.

Using the information in such a worksheet, WLM definitions can be created. Keep in mind that a table like this is a working document that is drafted, refined, and changed as more is learned about existing workloads, or new workloads are added to the environment.

3.3 Manage the work

The WLM management stage can be used to make steady progress toward meeting your business goal, and to identify actions that you can take when there are indications that the goal is not being met. Use the worksheet 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 create historical analysis reports.

3.3.1 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 3-2 illustrates the service classes we defined. The service classes address the WLM worksheet column **Action**, shown in Table 3-1 on page 54.

We could have altered the default service classes and workloads, but in this case we wanted 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.

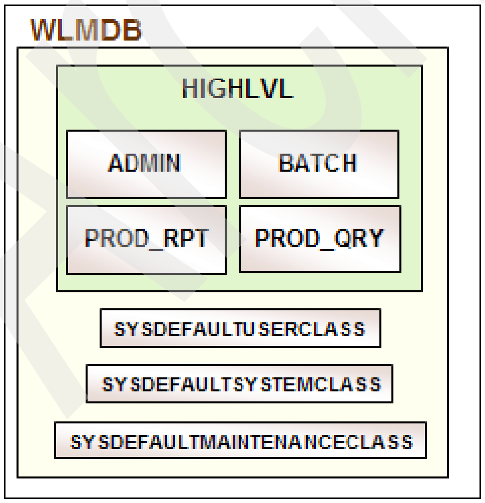


Figure 3-2 WLMDB service classes

Example 3-1 shows service classes DML. We chose 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, but *not* for the level for all service classes. The type of collection depends on the type of data to be collected. The WLM documentation explains each type of data collection. It is recommended that you collect only the data that will be used to monitor and report on the workloads.

Notice in the example that Aggregate Request Data is only collected for service class BATCH. For batch workloads, we want to monitor and report their average request execution times, which is collected for Aggregate Request Data.

Example 3-1 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: When setting up your initial configuration, 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.

3.3.2 Creating the workloads

Use the CREATE WORKLOAD statement to define the workloads and specify which subclass is responsible for handling the workload. The workloads address the WLM worksheet column **Identification**, shown in Table 3-1 on page 54. Each workload is tied to a subclass.

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

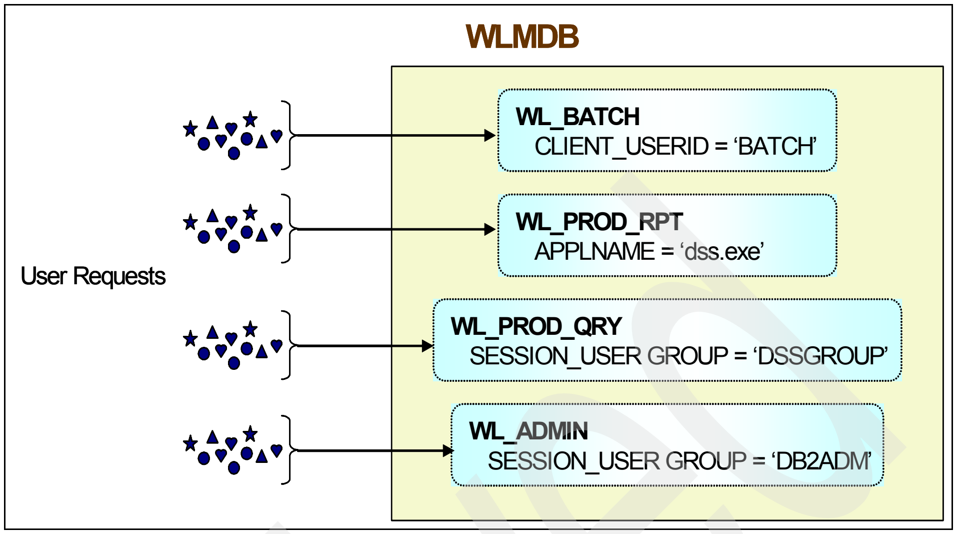


Figure 3-3 WLMDB workloads

Example 3-2 shows the workloads we set up, again, allowing the WLM setup to evolve. (Note that although SYSDEFAULTUSERCLASS, SYSDEFAULTSYSTEMCLASS, and SYSDEFAULTMAINTENANCECLASS are automatically created, to keep things simple they are not shown here.)

Example 3-2 Creating workloads

```
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_qry UNDER highlvl POSITION AT 3;

CREATE WORKLOAD wl_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. Do not 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 3-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 3-tier environment.

In our case, we want all batch jobs to be identified by the client_userid BATCH. Example 3-3 shows a batch job using the wlm_set_client_info stored procedure.

Example 3-3 Batch job using the wlm_set_client_info

```
db2 connect to wlmdb
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.

3.3.3 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 case, we granted all workloads to PUBLIC because there were no security or audit concerns in the test environment; see Example 3-4 on page 60.

Keep in mind, however, that granting to PUBLIC should be used with caution. Granting to PUBLIC in production environments could allow a knowledgeable user to run work in higher priority service classes by changing their client user ID, client application name, client workstation name, and client accounting string to match a higher priority service class.

Example 3-4   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, we enabled the service classes and workloads. As stated earlier, we recommend that you create all service classes and workloads using DISABLED in order to prevent premature usage until the entire WLM setup has been completed.

3.3.4 Creating the event monitor

The final step in our basic setup is to create the event monitor. Because we chose to collect aggregate statistics for historical analysis, an event monitor is needed.

As shown in Example 3-5, we decided to write the event monitor data to tables instead of files. Because the amount of data is expected to be small, we wanted the flexibility of tailoring our reports using SQL.

Example 3-5   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 3-6 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 were only performing aggregate collection, which uses the tables:

- SCSTATS
- WLSTATS
- HISTOGRANBIN

The other tables were created in case we needed them later. All the event monitor tables are created in a specific tablespace as a good administration practice. Placing the tables in a specific tablespace allows the DBA to determine their location, tablespace type, and the amount of space allocated so they potentially will not contend with production tables.

Example 3-6 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
```

3.3.5 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 work or take corrective actions, as required.

Because this workload is not affected by thresholds, however, 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;
```

Note that 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 that is constructed to delete the prior WLM configuration, an SQL4714N error may occur when all WLM service classes are disabled. The reason for this is because if all service classes have been disabled, nothing else in the script can execute until the work is routed to an enabled workload. So, this is where `SYSDEFAULTADMWORKLOAD` is useful.

Example 3-7 shows a full script used in the test environment.

Example 3-7 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, after all the service classes are disabled, the script cannot continue because we disabled every service class we are using. An SQL4714N appears when the command shown is attempted:

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

3.4 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**, shown in Table 3-1 on page 54. Are we meeting the business requirements that were documented previously in our worksheet?

In our case, we collect aggregate data over time and monitor it periodically. From the reports, we learn about our workloads and determine if additional action is needed. The details of monitoring are covered in Chapter 4, “Monitoring DB2 workload management information” on page 75. Here, we discuss only the reports used in our basic WLM setup.

Since we are set up 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 3-8 shows the export SQL statements.

Example 3-8 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))
  END AS avg_c_exe_tm,
  CASE WHEN 0 > INT(SUM(request_exec_time_avg))
       THEN 0
       ELSE INT(SUM(request_exec_time_avg))
  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
```

Note: Each column is summed because the SCSTATS table contains a row for each partition, but the columns used contain values from the coordinator partition, so the other partitions will have a zero (0) value and our query would contain an error and not complete successfully.

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 3-4 shows the requests execution time by subclass on a typical day.

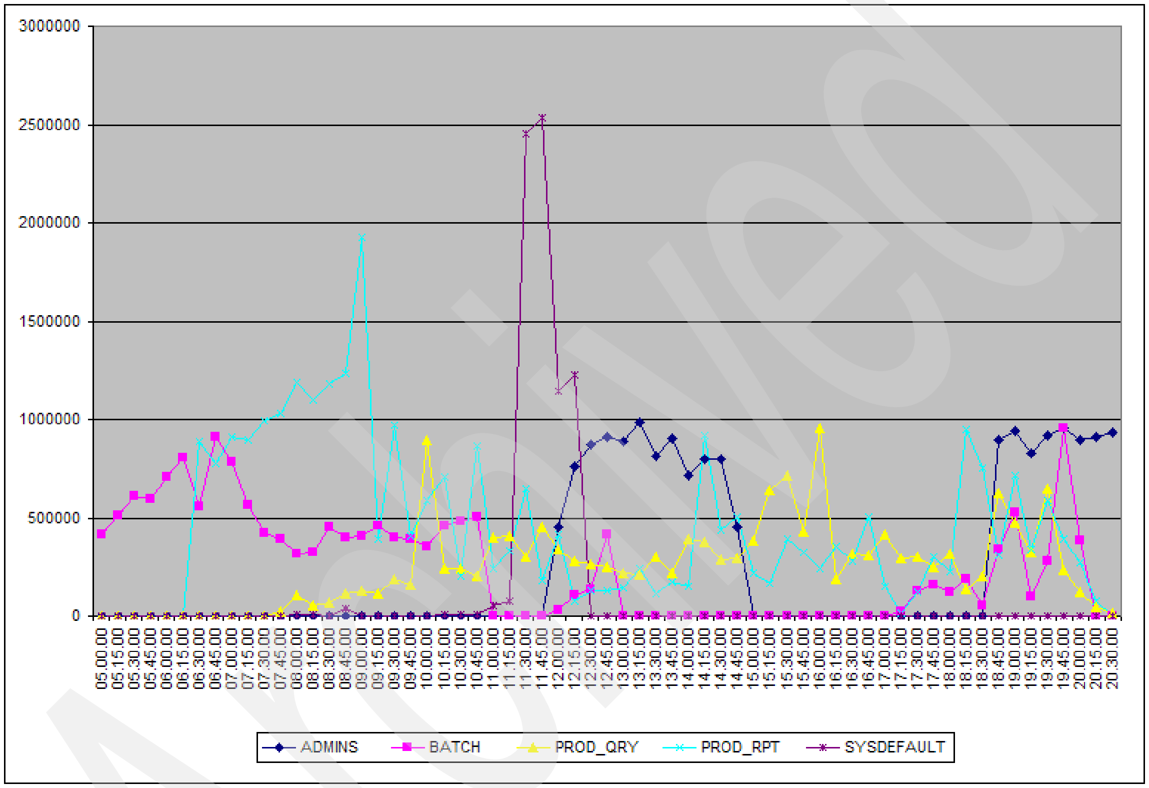


Figure 3-4 Request execution time by subclass

This graph shows 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 processes running between 12:00 pm and 1:00 pm.
- Production reports begin at 6:30 am and continue all during the prime shift, concluding 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 steadily until 8:00 pm. Again, we have users in several time zones. We also see heavy CPU loads at 9:00 am and 4:00 pm.
- The Admins appear to be running heavy loads twice a day, from 12 pm to 1:00 pm and from 6:00 pm to 8:30 pm. From our inquiry, we discover that the online table space backups are run during these times daily.
- We also see an unaccounted-for workload in the SYSDEFAULTSUBCLASS. Someone 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 effect of our workloads on the system as shown in Figure 3-5. 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.

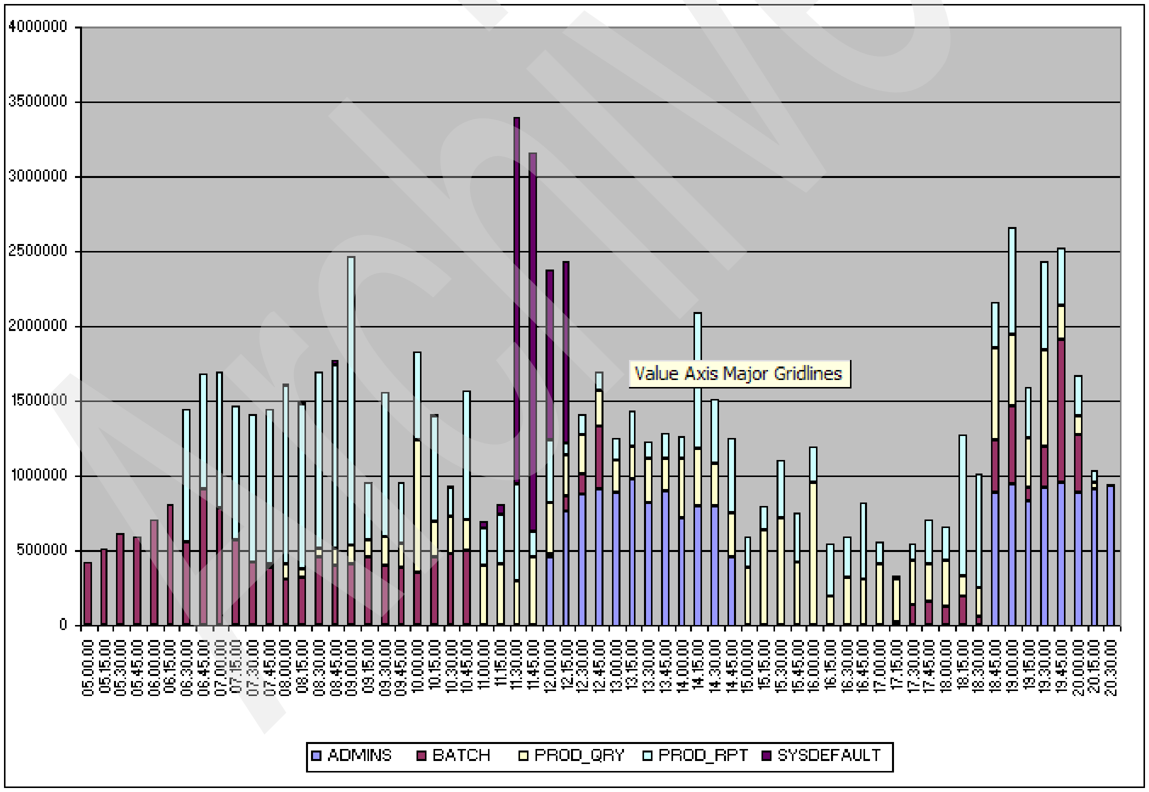


Figure 3-5 Bar chart - Request execution time by subclass

We now turn our attention to concurrent top connections to get a sense of the throughput of our work. This may not be an exact measurement of throughput, but it does show how many queries are concurrently running throughout the day.

With many concurrent queries running, more resources could be needed, such as memory, CPU, and higher disk activity. We get a sense of how many workloads are running during the day, as shown in Figure 3-6 on page 67.

Using the concurrent_top_connections, we can analyze how many queries are running concurrently throughout the day. We see from this graph that our production ad hoc queries concurrency usually peaks twice a day. Our batch jobs appear to taper off around 11 am.

We also see activity in the SYSTEMDEFAULTSUBCLASS during the prime shift between 11:00 am and 1:30 pm. As seen in Figure 3-4 on page 65, this has a severe impact on our CPU availability, and further investigation is warranted.

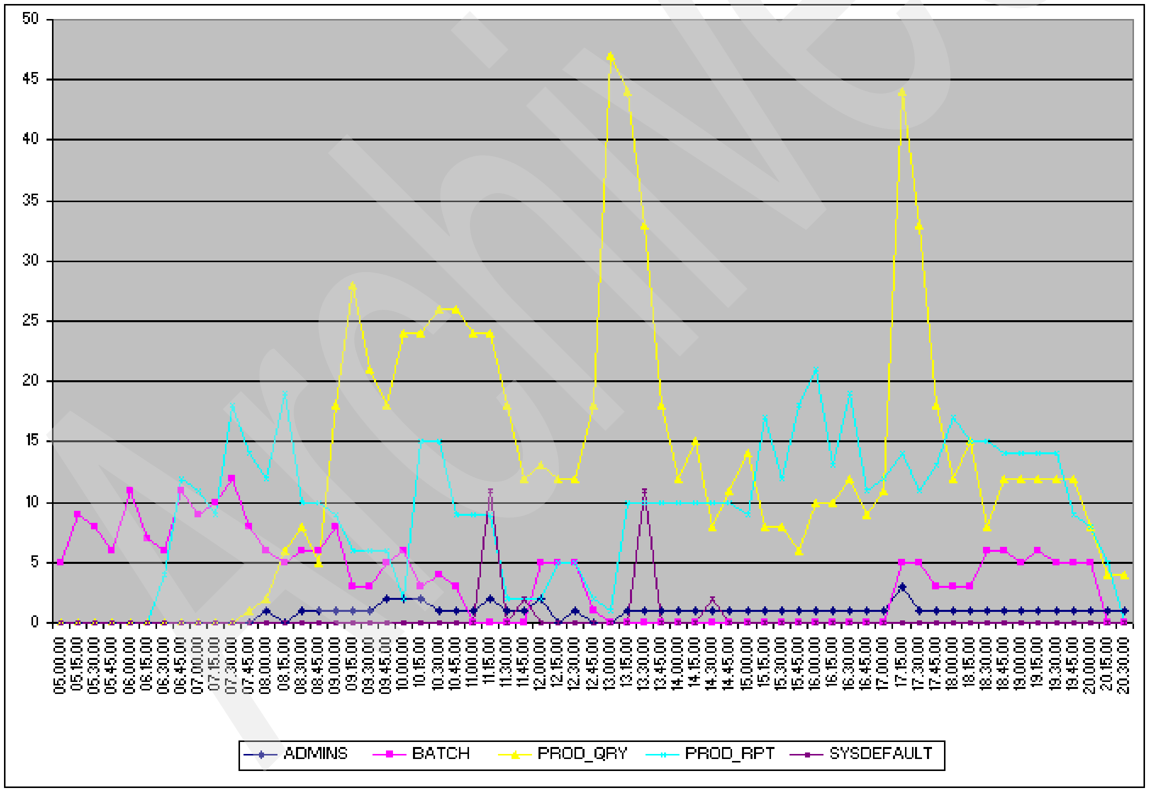


Figure 3-6 Active connections

Checking concurrent connections

Looking at the concurrent connections, we can get a picture of how much workload is contending for resources; see Figure 3-7 on page 68. (Again, this is not a complete picture, but rather a simplistic high level view.)

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. More analysis is need to confirm that suspicion.

Again, the SYSTEMDEFAULTSUBCLASS has connections during the peak periods of the day, as well as connections throughout most of the day. This could be causing problems and needs further investigation.

As you can see, then, through using a simple query and chart we can get a sense of what our system is doing at a high level and what, if any, additional areas need investigating.

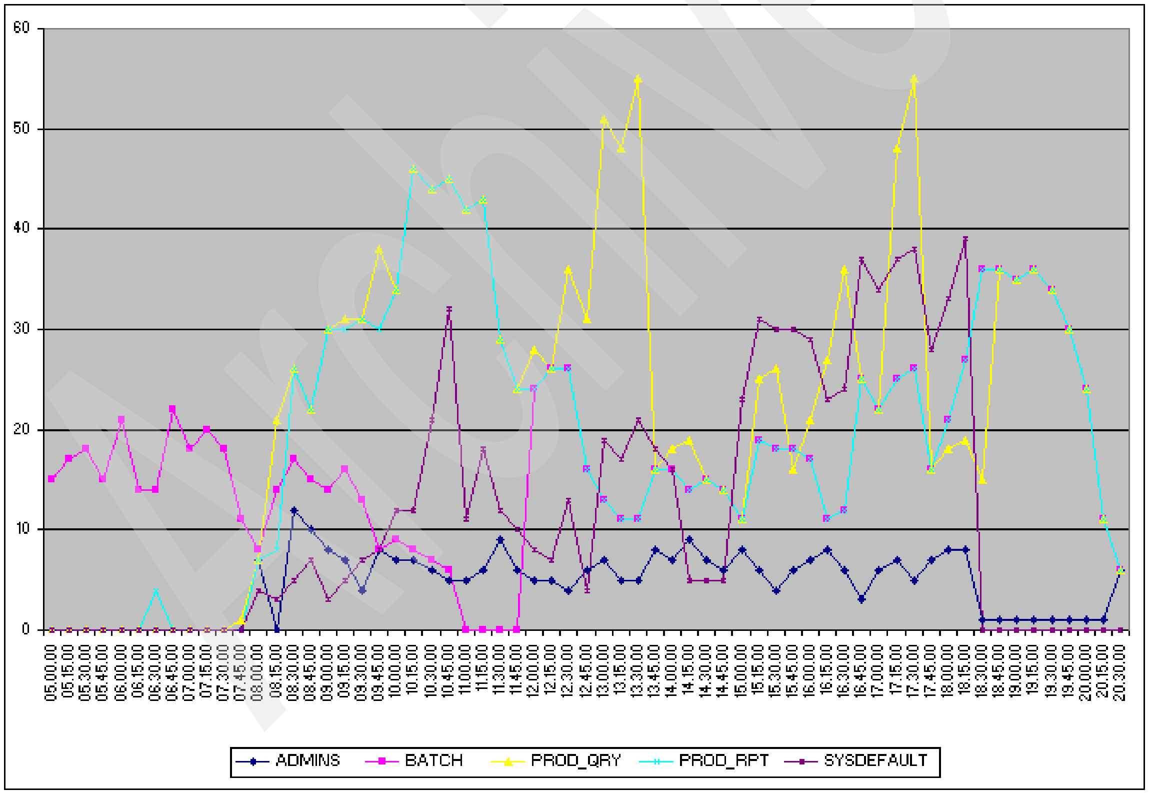


Figure 3-7 Concurrent connections

Restating the same information but as a stacked bar graph, we get an overall view of the number of connections hitting our system throughout the day, as shown in Figure 3-8 on page 69. We can see the high water mark for connections, and notice that it occurs twice a day during prime shift. We can also see which workloads are part of those peaks.

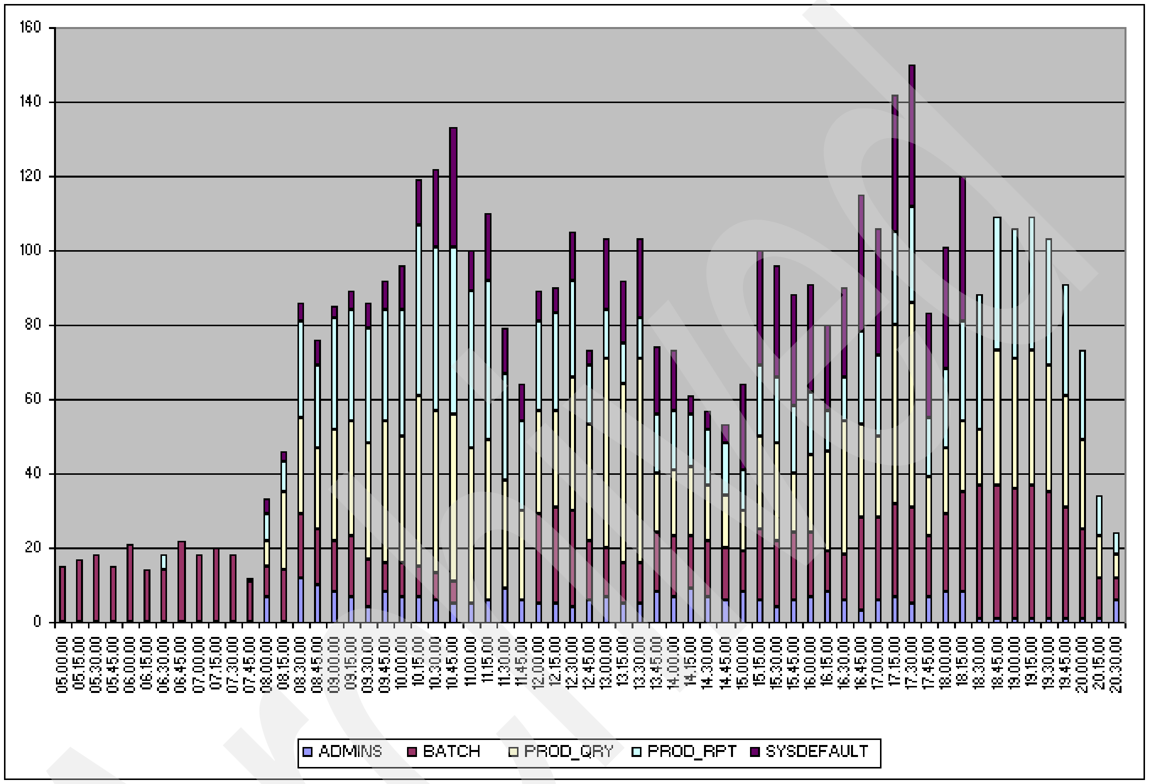
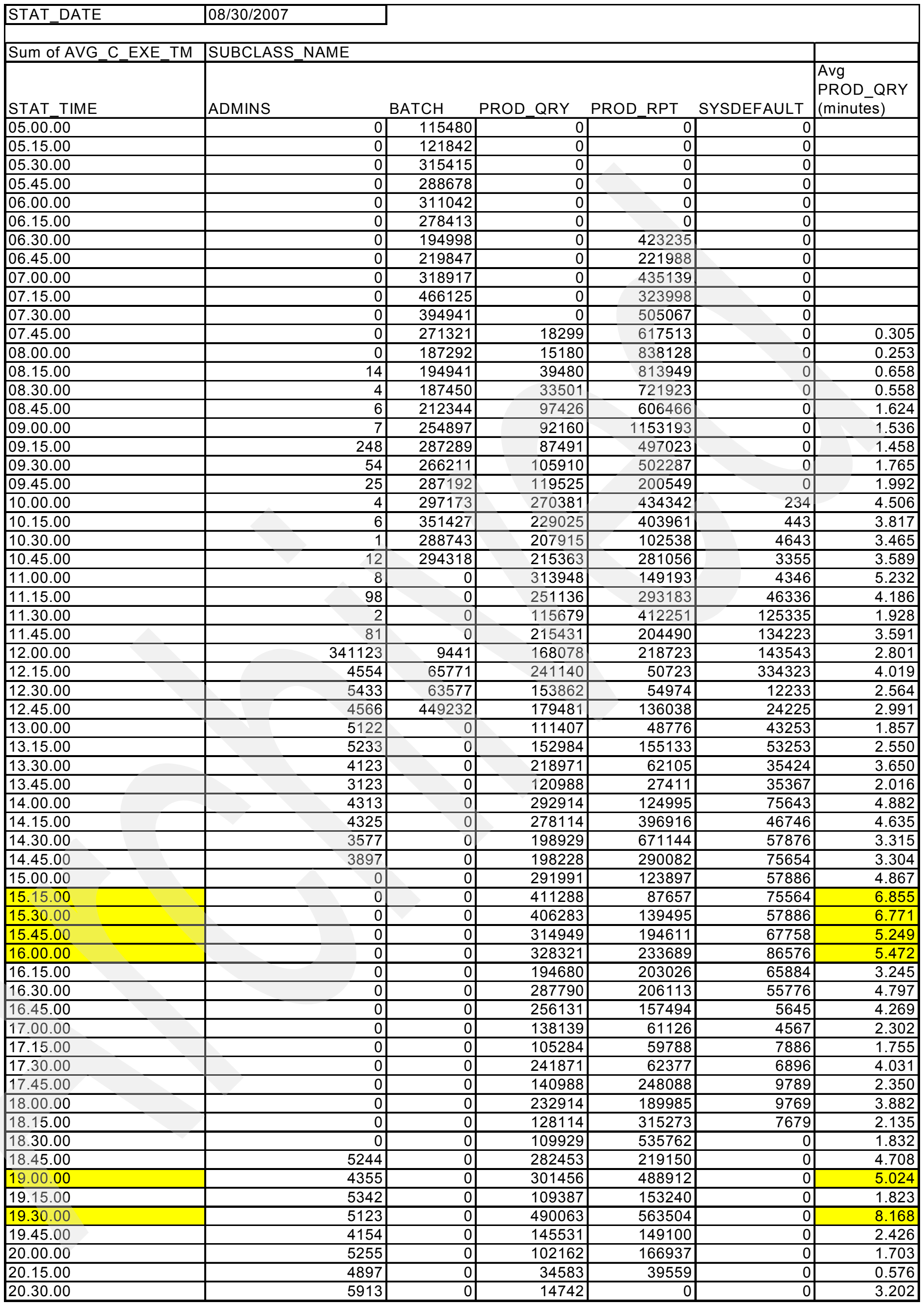


Figure 3-8   Connection high water mark

The spreadsheet shown in Figure 3-9 on page 70 displays the average execution times of the coordinators. Using this information, we can examine the average execution times for our workloads.

We want to know how long our ad hoc queries are averaging throughout the day to determine whether we are meeting our business requirement that 90% of the queries must complete within 5 minutes.



STAT_DATE	08/30/2007					
Sum of AVG_C_EXE_TM	SUBCLASS_NAME					
STAT_TIME	ADMINS	BATCH	PROD_QRY	PROD_RPT	SYSDEFAULT	Avg PROD_QRY (minutes)
05.00.00	0	115480	0	0	0	
05.15.00	0	121842	0	0	0	
05.30.00	0	315415	0	0	0	
05.45.00	0	288678	0	0	0	
06.00.00	0	311042	0	0	0	
06.15.00	0	278413	0	0	0	
06.30.00	0	194998	0	423235	0	
06.45.00	0	219847	0	221988	0	
07.00.00	0	318917	0	435139	0	
07.15.00	0	466125	0	323998	0	
07.30.00	0	394941	0	505067	0	
07.45.00	0	271321	18299	617513	0	0.305
08.00.00	0	187292	15180	838128	0	0.253
08.15.00	14	194941	39480	813949	0	0.658
08.30.00	4	187450	33501	721923	0	0.558
08.45.00	6	212344	97426	606466	0	1.624
09.00.00	7	254897	92160	1153193	0	1.536
09.15.00	248	287289	87491	497023	0	1.458
09.30.00	54	266211	105910	502287	0	1.765
09.45.00	25	287192	119525	200549	0	1.992
10.00.00	4	297173	270381	434342	234	4.506
10.15.00	6	351427	229025	403961	443	3.817
10.30.00	1	288743	207915	102538	4643	3.465
10.45.00	12	294318	215363	281056	3355	3.589
11.00.00	8	0	313948	149193	4346	5.232
11.15.00	98	0	251136	293183	46336	4.186
11.30.00	2	0	115679	412251	125335	1.928
11.45.00	81	0	215431	204490	134223	3.591
12.00.00	341123	9441	168078	218723	143543	2.801
12.15.00	4554	65771	241140	50723	334323	4.019
12.30.00	5433	63577	153862	54974	12233	2.564
12.45.00	4566	449232	179481	136038	24225	2.991
13.00.00	5122	0	111407	48776	43253	1.857
13.15.00	5233	0	152984	155133	53253	2.550
13.30.00	4123	0	218971	62105	35424	3.650
13.45.00	3123	0	120988	27411	35367	2.016
14.00.00	4313	0	292914	124995	75643	4.882
14.15.00	4325	0	278114	396916	46746	4.635
14.30.00	3577	0	198929	671144	57876	3.315
14.45.00	3897	0	198228	290082	75654	3.304
15.00.00	0	0	291991	123897	57886	4.867
15.15.00	0	0	411288	87657	75564	6.855
15.30.00	0	0	406283	139495	57886	6.771
15.45.00	0	0	314949	194611	67758	5.249
16.00.00	0	0	328321	233689	86576	5.472
16.15.00	0	0	194680	203026	65884	3.245
16.30.00	0	0	287790	206113	55776	4.797
16.45.00	0	0	256131	157494	5645	4.269
17.00.00	0	0	138139	61126	4567	2.302
17.15.00	0	0	105284	59788	7886	1.755
17.30.00	0	0	241871	62377	6896	4.031
17.45.00	0	0	140988	248088	9789	2.350
18.00.00	0	0	232914	189985	9769	3.882
18.15.00	0	0	128114	315273	7679	2.135
18.30.00	0	0	109929	535762	0	1.832
18.45.00	5244	0	282453	219150	0	4.708
19.00.00	4355	0	301456	488912	0	5.024
19.15.00	5342	0	109387	153240	0	1.823
19.30.00	5123	0	490063	563504	0	8.168
19.45.00	4154	0	145531	149100	0	2.426
20.00.00	5255	0	102162	166937	0	1.703
20.15.00	4897	0	34583	39559	0	0.576
20.30.00	5913	0	14742	0	0	3.202

Figure 3-9 Average coordinator execution times

3.5 Summary

Now that we completed our cycle for the first time, what did we learn? Looking back at our business requirements we see that not all of them are being met.

Batch ETL must complete prior to primetime (8 am), but we see from our charts that batch runs past 8 am. In fact, batch appears to run until around 1 pm.

Ad hoc queries must compete 90% in under 5 minutes. If we assume the coordinator execution time is representative of the overall query execution time, we can extrapolate the average query execution time as 3.2 minutes, with only 6 time periods with an average execution time greater than 5 minutes (15:15, 15:30, 15:45, 16:00, 19:00, and 19.30). This appears to be within our business requirement.

Production reports are completing but they start at 6:30 am and finish at 20:15 pm. So, we need to see why reports are being started so early.

We also see that there are workloads outside of our definition that need to be more explicitly identified.

The analysis of the data will be the input for the next cycle using our WLM methodology, as discussed in the following sections.

Identify

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

- Identify what is running in the SYSTEMDEFAULTSUBCLASS. We have already identified all existing categories. Therefore, this work is probably being performed by our report development group because it is the only remaining group allowed access to our example database.
- Limit the resources for the rouge queries based on execution time
- The BATCH group runs two types of workloads, LOAD and our ETL tool (etl.exe). We want to identify how much resource is being used for each of these categories.

Manage

To reach the new goals we identified, we can alter our setup to the one shown in Example 3-9. The alterations will drop the original BATCH service class and its related workload and create two new services classes and workload in its place.

This will further divide the BATCH service class into ETL and LOADS for more specific monitoring and reporting. Because aggregate request data is being requested, we can monitor and report on average request execution times to see how long these jobs run.

A new service class is also created to monitor and report on the activity of the development group's activity and impact on the data warehouse. Using the additional information, we may want to limit the number of queries, the amount of resource, or both, that they can consume.

Example 3-9   Altered basic setup

```
ALTER WORKLOAD wl_batch DISABLE;

ALTER SERVICE CLASS BATCH UNDER highlvl DISABLE;

DROP WORKLOAD wl_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 wl_batch_etl APPLNAME  ('etl.exe') DISABLE SERVICE CLASS
batch_etl UNDER highlvl POSITION AT 2;
CREATE WORKLOAD wl_batch_load CURRENT CLIENT_USERID ('BATCH') DISABLE SERVICE
CLASS batch_load UNDER highlvl POSIION AT 3;
ALTER WORKLOAD wl_prod  POSITION AT 4;
ALTER WORKLOAD wl_admin  POSITION AT 5;
CREATE WORKLOAD wl_dev_rpt SESSION_USER GROUP ('DEVGRP')  DISABLE SERVICE CLASS
HIGHLVL POSITION AT 6;

GRANT USAGE ON WORKLOAD wl_batch_etl TO PUBLIC;
GRANT USAGE ON WORKLOAD wl_batch_load TO PUBLIC;
GRANT USAGE ON WORKLOAD wl_dev_rpt TO PUBLIC;

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

ALTER WORKLOAD wl_batch_etl ENABLE;
ALTER WORKLOAD wl_batch_load ENABLE;
ALTER WORKLOAD wl_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.

4

Monitoring DB2 workload management information

This chapter describes the methodology for monitoring the DB2 workload management information. We discuss the monitoring tools in two categories: real-time monitoring and historical monitoring. For information about the concept, and to learn event monitor-related terminology, refer to the DB2 Information Center:

`http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/`

We discuss the following topics in this chapter:

- 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

4.1 Real-time monitoring

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

4.1.1 Workload management table functions

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

These table functions offer you the ability to access monitoring data (such as workload management statistics) by using SQL statements. You can write applications to query and analyze data as if it were any physical table on the data server.

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 active database partitions.

As described next, 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 returns information about work that is currently executing on the system.

- WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES

  Table function parameters:

  - service_superclass_name
  - service_subclass_name
  - dbpartitionnum

  You can also use this to list all workload occurrences on the system if the optional service class parameters are not specified. 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 4-1 shows how to use this table function to find out which applications are connected to the system.

Example 4-1 Workload occurrences 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, SUBSTR(application_name,1,10) AS
application_name, SUBSTR(client_applname,1,10) AS client_applname
FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('','',-2))"

SUPERCLASS_NAME     SUBCLASS_NAME       WORKLOAD_NAME
APPLICATION_HANDLE   WORKLOAD_OCCURRENCE_STATE
  APPLICATION_NAME CLIENT_APPLNAME
------------------- ------------------- ----------------------
-------------------- -------------------------------
- ---------------- ---------------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS  SYSDEFAULTUSERWORKLOAD
9 UOWEXEC
  db2bp.exe        test
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS  SYSDEFAULTUSERWORKLOAD
53 UOWWAIT
  java.exe         account

  2 record(s) selected.
```

Note that the WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function does not report applications that have status “connected” because those applications have not been assigned to a workload yet.

- 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. For example, if someone complains about a query not running, you can find the workload occurrence and see what agents are working on the query and what state they are in.

Example 4-2 lists the agents currently working on behalf of the applications identified by application handle 9 and 53.

Example 4-2 Using WLM_GET_SERVICE_CLASS_AGENTS

```
>db2 "SELECT application_handle, SUBSTR(event_type,1,10) AS event_type,
SUBSTR(event_object,1,10) AS event_object, SUBSTR(event_state,1,10) AS
event_state, SUBSTR(request_type,1,10) AS request_type
FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('', '', 9, -1)) AS agents"

APPLICATION_HANDLE   EVENT_TYPE EVENT_OBJECT EVENT_STATE REQUEST_TYPE
-------------------- ---------- ------------ ----------- ------------
                   9 PROCESS    ROUTINE      EXECUTING   OPEN

  1 record(s) selected.

>db2 "SELECT application_handle, substr(event_type,1,10) as event_type,
substr(event_object,1,10) as event_object, substr(event_state,1,10) as
event_state, substr(request_type,1,10) as request_type
FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('', '', 53, -1)) AS Agents"

APPLICATION_HANDLE   EVENT_TYPE EVENT_OBJECT EVENT_STATE REQUEST_TYPE
-------------------- ---------- ------------ ----------- ------------
                  53 WAIT       REQUEST      IDLE        COMMIT

  1 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.

  Example 4-3 shows all activities on the system from all applications. Two applications are executing a SELECT, VALUES INTO, or XQuery statement.

Example 4-3 Activities on the system from all applications

```
>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.
```

Example 4-4 shows the total number of LOADs currently running on the system.

Example 4-4 Number of LOAD activities on the system

```
>db2 "select count(*) as load_count from
table(wlm_get_workload_occurrence_activities(cast(NULL as bigint), -1))
where activity_type='LOAD'"

LOAD_COUNT
-----------
          5

  1 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 time stamp switch to collect some elements (for example, CPU times and rows read or modified). The value -1 means that either the statement monitor switch or the time stamp switch is not activated.

  Example 4-5 shows the details for the application with handle 18. It shows details about the activity with unit of work ID 16 and activity ID 1.

Example 4-5 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 a SQL statement in lock-wait

In this section, we provide a simple example showing how the table functions can be used. If you want to get the SQL statement that is in lock-wait by using the snapshot monitor, you need to turn on the statement monitor switch. WLM table functions provide you with a way to obtain the running SQL statement without turning the monitoring switch on, because it does not depend on the monitor switch.

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 4-6. The application handle 43 is lock-waited.

Example 4-6 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              db2wlmd
                  47 UOWEXEC                db2bp.exe
                  40 CONNECTED              db2taskd

  7 record(s) selected.
```

2. Identify the UOW ID and the activity ID using the WLM_GET_SERVICE_CLASS_AGENTS table function. Example 4-7 shows UOW ID and activity ID for application 43, respectively.

Example 4-7 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.
```

3. Identify the SQL statement using the WLM_GET_ACTIVITY_DETAILS table function. Example 4-8 shows the query in lock-wait in `STMT_TEXT`. The `STMT_TEXT` field only contains the first 1024 characters of the statement text.

Example 4-8 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 returns the detailed workload information since the last time that the statistics were reset. These table functions report only a subset of the statistics event monitor. To view the full set of statistics, you must collect the statistics information. We discuss what statistics are captured and stored in

4.2.3, “Statistics event monitor” on page 107, and explain how they are reset in “Resetting statistics on workload management objects” on page 110.

Note: The monitoring table functions and stored procedures are activities (like any other SQL) and as such are subject to monitoring. For example, if they are run under a given service class, they will show up in the completed activity counts for that service class.

If you want the statistics from user applications isolated from the statistics from monitoring activities, consider isolating the user applications from the monitoring activities; that is, run them in different service classes.

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, knowing the high water marks for concurrent connections is useful when determining peak workload activity.

  Example 4-9 shows the concurrent connections for each service superclass.

  Example 4-9 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 the number of activities and average execution times. The average execution time is useful to know when looking at general system health and the distribution of activities across service classes and partitions.

Some statistics (for example, average activity lifetime) are only returned if aggregate activity data collection (specified using the COLLECT AGGREGATE ACTIVITY DATA clause) is enabled for the service subclass. Other statistics (such as request execution time) depend on whether or not aggregate request data collection (specified using the COLLECT AGGREGATE REQUEST DATA clause) is enabled.

Example 4-10 shows the number of requests (NUM_REQUESTS_ACTIVE) that are executing in the service subclass and the average request execution time (REQUEST_EXEC_TIME_AVG). Notice that for all service subclasses the average request execution time is NULL. This is because the COLLECT AGGREGATE REQUEST DATA clause has not been specified for any of these service subclasses.

Example 4-10 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 of each work class that had a work action from the corresponding work action set applied to them. 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, we are using the work action set and work class to determine the number of large read activities running. The work_class_name marked with an asterisk (*) represents all activities that did not fall into the explicitly named work class LARGEREADS_QUERY. The value 3 means that three activities of type `LARGEREADS_QUERY` had work actions applied to them from the `LARGEREADS_ACTIONSET` work action set.

Example 4-11 Using WLM_GET_WORK_ACTION_SET_STATS

```
db2 "SELECT substr(work_action_set_name,1,20) as work_action_set_name,
substr(work_class_name,1,16) 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
-------------------- ---------------- --------- --------------------------
LARGEREADS_ACTIONSET  *                0        2007-08-24-18.22.02.038311
LARGEREADS_ACTIONSET  LARGEREADS_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. Knowing these is useful when you are monitoring general system health or drilling down to identify problem areas.

  Example 4-12 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 4-12 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 include 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. Very little queuing might indicate that a threshold is not restrictive enough, or is not needed.

  Example 4-13 shows the CPU time consumed by each service class. From the output we can learn which service class is consuming the most CPU resources.

Example 4-13 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

The snapshot monitor is also useful for capturing the condition of a 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 4-1 lists the fields shared between the workload management table functions and the snapshot monitor table functions.

Table 4-1 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

Example 4-14 shows how to identify the CPU consumption of applications by joining the WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function and the SNAPAPPL administrative view. In this example, we see that the `ACCOUNTING` superclass consumes the most CPU resources.

Example 4-14 Identifying applications consuming CPU resource

```
SELECT substr(wlm.service_superclass_name,1,25) as superclass_name,
substr(wlm.service_subclass_name,1,25) as subclass_name, sum(s
nap.agent_usr_cpu_time_s + snap.agent_sys_cpu_time_s) as cpu_time FROM
sysibmadm.snapappl snap, TABLE(WLM_GET_SERVICE_CLASS_WORKLO
AD_OCCURRENCES('','',-1)) wlm WHERE snap.agent_id=wlm.application_handle GROUP
BY wlm.service_superclass_name, wlm.service_subclas
s_name ORDER BY cpu_time DESC

SUPERCLASS_NAME           SUBCLASS_NAME             CPU_TIME
------------------------- ------------------------- --------------------
ACCOUNTING                SYSDEFAULTSUBCLASS                          47
SALES                     SYSDEFAULTSUBCLASS                          39
SYSDEFAULTUSERCLASS       SYSDEFAULTSUBCLASS                           0

  3 record(s) selected.
```

4.1.2 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. Cancelling an activity will not terminate the connection. It only stops the activity that was cancelled. If an application has several activities running and only one needs to be stopped, cancelling the activity is a gentler approach than forcing the application, which would terminate all activities. The application with the canceled activity receives the error SQL4725N.

  In Example 4-15, from window A, we first tried to cancel application 637 with UOW ID 3 and ACTIVITY ID 1. This is not an existing activity, however, and DB2 returns SQL4702N with SQLSTATE 5U035. We then successfully cancelled application 637, UOW ID 1, ACTIVITY ID 1. In WindowB, we see the application is canceled but the connection stays.

Example 4-15 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  FIRSTNME     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 queued or 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, SQL1633W with SQLSTATE 01H53 is returned.

  Example 4-16 shows how to use this stored procedure to collect the estimated cost for a particular activity from the active activities event monitor. SQL1633W is returned on the first stored procedure call because the event monitor was not activated. After the event monitor was activated, the stored procedure ran successfully and we could select data from the active activities event monitor.

  For more information about the activities event monitor, refer to the 4.2.1, "Activities event monitor" on page 95.

Example 4-16 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
```

```
>db2 "select query_card_estimate, query_cost_estimate from
activity_db2activities as act where act.agent_id = 660 and act.uow_id=2 and
act.activity_id = 1"

QUERY_CARD_ESTIMATE  QUERY_COST_ESTIMATE
-------------------- --------------------
                   1              1530286

  1 record(s) selected.
```

- 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; they are not collected.

  Example 4-17 shows that the data of last_reset field is changed after calling the WLM_COLLECT_STATS stored procedure. The evmon statistical tables show that these tables contain the statistics that previously were being reported by the table function. For more information about the statistics event monitor, refer to "Statistics event monitor" on page 107.

Example 4-17 calling the WLM_COLLECT_STATS stored procedure

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

SERVICE_SUPERCLASS_NAME CONCURRENT_CONNECTION_TOP LAST_RESET
----------------------- ------------------------- --------------------------
SYSDEFAULTUSERCLASS     4                         2008-01-01-19.34.00.895855

1 record(s) selected.

>db2 "CALL WLM_COLLECT_STATS()"

  Return Status = 0

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

SERVICE_SUPERCLASS_NAME CONCURRENT_CONNECTION_TOP LAST_RESET
----------------------- ------------------------- --------------------------
SYSDEFAULTUSERCLASS     4                         2008-01-01-19.34.56.354245
```

```
  1 record(s) selected.

>db2 "select substr(service_superclass_name,1,20) as
service_superclass_name, concurrent_connection_top, last_wlm_reset,
statistics_timestamp from scstats_db2statistics where
service_superclass_name='SYSDEFAULTUSERCLASS'"

SERVICE_SUPERCLASS_NAME CONCURRENT_CONNECTION_TOP LAST_WLM_RESET
STATISTICS_TIMESTAMP
----------------------- ------------------------- --------------------------
--------------------------
SYSDEFAULTUSERCLASS     4                         2008-01-01-19.23.18.674420
2008-01-01-19.34.00.895855
SYSDEFAULTUSERCLASS     4                         2008-01-01-19.34.00.895855
2008-01-01-19.34.56.354245

  2 record(s) selected.
```

- WLM_SET_CLIENT_INFO

  Syntax:

  WLM_SET_CLIENT_INFO(client_userid, client_wrkstnname, client_applname, client_acctstr, client_workload)

  You can use this procedure to set client information (client's user ID, application name, workstation name, accounting, or workload information) associated with the current connection to the DB2 server.

  Example 4-18 shows how to use this stored procedure to set the client user ID.

Example 4-18 Setting the client's user ID

```
>db2 CALL SYSPROC.WLM_SET_CLIENT_INFO('V95USER', NULL, NULL, NULL, NULL)
  Return Status = 0

>db2 values(current client_userid)
1
--------------------------------------------------------------------------
V95USER

  1 record(s) selected.
```

4.1.3 The db2pd command for workload management

In DB2 9.5, the **db2pd** command is enhanced for workload management to return operational information from the DB2 database system memory sets. The **db2pd** command allows you to display current state information about WLM objects.

Here we list the **db2pd** command with various options for workload management:

- db2pd -workloads [*none/workloadID*]

  This returns a list of workload definitions in memory at the time the command is run.
- db2pd -serviceclasses [*none/serviceclassID*]

  This 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*]

  This 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*]

  This 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 -thresholds [*none/thresholdID*]

  This 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 4-19 shows how we used **db2pd** to retrieve information for service class ID 3 in SAMPLE database. You can see the application handle list associated with the service class.

Example 4-19 Using the 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
```

Example 4-20 shows how we used **db2pd** to look at the thresholds to see in which queue this agent is actually waiting. You can see the application handle 97 is queuing the QUEUE_THRESH threshold.

Example 4-20 Check thresholds with db2pd

```
>db2pd -db sample -thresholds

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:36:13

Database Thresholds:

Service Class Thresholds:

Threshold Name              = QUEUE_THRESH
Threshold ID                = 1
Domain                      = 30
Domain ID                   = 3
Predicate ID                = 90
Maximum Value               = 50
Enforcement                 = D
Queueing                    = Y
Queue Size                  = 0
Collect Flags               = N
Partition Flags             = C
Execute Flags               = C
Enabled                     = Y

Database Threshold Queues:
```

```
Service Class Threshold Queues:

Queue information for threshold: QUEUE_THRESH
Max Concurrency            = 50
Concurrency                = 50
Max Queue Size             = 0

Agents Currently Queued:
EDU ID       AppHandl [nod-index]         Agent Type  Activity ID  UOW ID
660          79       [000-00079]  1            1            1
```

4.2 Historical monitoring

Historical monitoring is useful for seeing what have occurred on your system. DB2 provides three event monitor types, ACTIVITIES, THRESHOLD VIOLATIONS and STATISTICS, for WLM. ACTIVITIES and STATISTICS event monitors store activity information or aggregate information. The THRESHOLD VIOLATIONS event monitor records threshold violations. This section describes how to collect and analyze this information.

Event monitors are used to collect specified event monitor data while they are active. Typically, event monitors write data to either tables or files. The event monitor file output requires formatting using db2evmon in order to be readable. The event monitor table data can be queried using SQL statements.

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

The following high level procedure is an example of using an event monitor.

1. Create an event monitor using the CREATE EVENT MONITOR statement.

   The event monitor can write output to the following:

   – Tables ([WRITE TO TABLE]): Event monitor tables are automatically created when creating a event monitor.

   – A file ([WRITE TO FILE]): Event files are in a directory location to be stored.

2. Activate an event monitor using the SET EVENT MONITOR STATE statement.

   ```
   SET EVENT MONITOR <event monitor name> STATE [1(ACTIVATE) | 0(DEACTIVATE]
   ```

3. Applications are run to collect their event monitor data.
4. Deactivate an event monitor using the SET EVENT MONITOR STATE statement.
5. Refer to event monitor data.
   - Tables: You can access this data by using SQL SELECT statement.
   - A file: You can format this data by using the **`db2evmon`** command.

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

Table 4-2 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 monitors ACTIVITIES, STATISTICS, or THRESHOLD VIOLATIONS can be active at any time. If an event monitor of the same type is already active, SQL1631N with SQLSTATE 5U024 is returned.

For detailed event monitor information, refer to the DB2 Information Center:

https://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.luw.admin.mon.doc/doc/c0005721.html

4.2.1 Activities event monitor

This section explains how to use the activities event monitor. The following are a few frequently asked questions about the activities event monitor:

- 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.

- How is this information useful?
  - This information can be useful as input to tools such as Design advisor(db2advis) or to the explain utility (to obtain access plans)
  - It can also help you to 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 the activity belongs, or by specifying a threshold that might be violated by such an activity.

Collecting WLM object activity information

Follow these steps to collect 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 4-21 shows how to create an activities event monitor that writes data to a file.

   Example 4-21 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 4-22 shows to how to create an activities event monitor that writes data to tables. Four tables are created.

Example 4-22 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.
```

To obtain a quick reference about the column name or the type name for those event monitor tables, use the DESCRIBE TABLE command:

```
db2 describe table activity_db2activities
```

For more details about the information collected in the table, refer to the Information Center:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/

2. Activate the event monitor.

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

   Example 4-23 shows how to set the event monitor to active and check its status.

Example 4-23 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 statements, and specify the COLLECT ACTIVITY DATA keywords.

   Example 4-24 shows how to alter the default service class to specify the COLLECT ACTIVITY DATA keywords. 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 4-24 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 activity 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 DETAILS option)

  This level collects detailed information, including statement text (static and dynamic SQL) and compilation environment.
- Detailed with input data values (COLLECT ACTIVITY DATA WITH DETAILS AND VALUES option)

  This level collects default information, detailed information, and data values.

In the following examples, we demonstrate the differences between these three levels.

In Example 4-25, we create service class, subclasses, and workload to collect Java™ application activity information. Three CREATE WORKLOAD statements are shown for collecting different level details of the activity. Each time, only one of the statements is run. We also create an activity event monitor that writes data to a file.

Example 4-25 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 4-26 shows the output of the activities event monitor for workload ACTWORKLOAD with the COLLECT ACTIVITY DATA WITHOUT DETAILS option.

Example 4-26 COLLECT ACTIVITY DATA WITHOUT DETAILS 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 4-27 shows the output of the activities event monitor for workload `ACTWORKLOAD` with the COLLECT ACTIVITY DATA WITH DETAILS option. The activity statement section is added. From `Statement text`, we can identify which query is being run by `Appl Handle 352`.

Example 4-27   COLLECT ACTIVITY DATA WITH DETAILS 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 4-28 shows the output of the activities event monitor for workload `ACTWORKLOAD` with the COLLECT ACTIVITY DATA WITH DETAILS 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 cannot collect the value of parameter marker.

Example 4-28 COLLECT ACTIVITY DATA WITH DETAILS 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 the 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 exists and activity information is stored. You cannot 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 must 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 by using the **db2advis** command, for example:

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

This example shows how to import activities collected by the DB2ACTIVITIES activity event monitor for the ACTWORKLOAD workload in the SAMPLE database into the design advisor. You can specify the workload or service class name, and the start time and end time.

4.2.2 Threshold violations event monitor

This section explains how to use the threshold violations event monitor. The following are frequently asked questions about the threshold violations event monitor:

- Which WLM objects can I collect threshold violation information for?

  This event monitor collects information about threshold violations, not the information about individual activities. Part of the violation information includes the identity of the activity that violated the threshold.

- Why is the information useful?

  Threshold violation information is useful for understanding the effectiveness of thresholds (for example, are they stopping too many activities?), learning whether any activities are violating thresholds, and identifying which activities violated thresholds so that you can investigate those activities further.

- How do I collect the information?

  When a threshold is violated, if there is an active threshold violations event monitor, a threshold volcanos record is written to this event monitor. This record identifies which threshold was violated, the time of the violation, the activity that violated the threshold, and the action taken (whether activity was allowed to continue or stop).

  If you need to know more information about the activity that violated the threshold, specify the COLLECT ACTIVITY DATA clause for the threshold. If COLLECT ACTIVITY DATA is specified for the threshold, detailed information about the activity that violated the threshold will be sent to the active activities event monitor when the activity completes.

  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?

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

Collecting threshold violations

Use the following steps to collect threshold violations.

1. Create an event monitor.

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

Example 4-29 shows how to create a threshold violations event monitor that writes data to a file.

Example 4-29 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 4-30 shows how to set the event monitor active.

   Example 4-30 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 4-31 shows how to create a threshold and specify the COLLECT ACTIVITY DATA keyword.

   Example 4-31 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 ran an SQL statement that should return over 10000 rows. Due to the threshold restriction, our query was stopped after receiving 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 4-32 shows the command and output of formatting the threshold violations event monitor.

Example 4-32 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 4-33 shows the command and output of formatting the activity event monitor. In our example, we only collected default activity information that included 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 4-33 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
```

4.2.3 Statistics event monitor

This section explains how to use the statistics event monitor and histograms. 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 aggregate activity information instead of individual activities.

The following are frequently asked questions about the 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:

  – At regular intervals that are specified by the WLM_COLLECT_INT database configuration parameter.

  – When 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
  - Threshold queues:
    - queue_assignments_total
    - queue_size_top
    - queue_time_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 you want to specify both a COLLECT AGGREGATE ACTIVITY keyword and a COLLECT AGGREGATE REQUEST DATA keyword, you can alter a service subclass to add the option.

Example 4-34 shows how to specify both keywords. In the first statement result, we know that the `PROD_QRY` service class already specifies the COLLECT AGGREGATE ACTIVITY DATA EXTENTED keyword. After altering the object to add the COLLECT AGGREGATE REQUEST DATA BASE keyword, the `COLLECTAGGREQDATA` value is changed to `B`.

Example 4-34 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 resets the statistics. If one user calls the stored procedure, a reset of the workload management 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.

The table functions return the in-memory statistics. So when those statistics are reset (through one of these events), the table functions will be reporting the reset values. You can determine when the statistics were last reset by looking at the last statistics reset time stamp that is included as part of each table function.

Automatic collecting statistics in a specific timeframe

Use the following steps to automatically collect statistics for a given workload management object in a given timeframe.

1. Create and activate a statistics event monitor.

   Use the CREATE EVENT MONITOR statement to create a STATISTICS event monitor as shown in Example 4-35. This event monitor writes data to file.

   Example 4-35 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 4-36.

   Example 4-36 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.

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

Example 4-37 shows how to alter the default service class to specify the COLLECT AGGREGATE ACTIVITY DATA BASE keyword.

Example 4-37 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 4-38 shows how to update the database configuration parameter.

   In this case, statistics information is sent to the statistics event monitor every five minutes. After you perform the preceding steps, workload management statistics are written to the statistics event monitor every wlm_collect_int minute.

   Example 4-38 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
```

   We used five minutes, simply for demonstration purposes. However, a short interval like this can generate too much data. For real usage, you should use a larger number.

4. Run your activities.
5. Format the statistics event monitor file.

   Example 4-39 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 4-39 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 histogram 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).

You create histogram templates to specify the uppermost bound for the range of data that the corresponding histogram captures. After you create histogram templates, you use these templates to create histograms of statistical data about database activity lifetimes for monitoring purposes. If you do not create histogram templates, the Design Studio uses a default template to create your histograms.

Histogram templates use multiple bins that individually collect the histograms from each database partition. The histograms in each bin are combined to create one histogram for all of the partitions in the database. You can use this histogram to express statistics such as the average response time per activity for a partition, the standard deviation of the response times for a partition, and the variance of the response times for a partition.

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 4-40 shows the histogram section of a statistics event monitor output. Three activities `Number in bin` have an execution time in the range zero (`Bottom`) milliseconds to one (`Top`) milliseconds.

Example 4-40 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 two reasons to create or alter a histogram template.

- Templates are used to get better understanding of distribution.

  For example, if you found all your activities were short and were falling into the first 10 bins, you may want to create a histogram template with a smaller high bin value to get a better sense of histogram distribution.

- Templates are used to create different histograms with different characteristics.

  For example, you would use a different template for an estimated cost histogram than for a time-based histogram.

Example 4-41 shows how to create a histogram template LIFETIMETEMP.

Example 4-41 Creating a histogram template

```
>db2 "create histogram template lifetimetemp high bin value 100000"
DB20000I  The SQL command completed successfully.

>db2 "select binid,binuppervalue from syscat.histogramtemplatebins where
templatename='LIFETIMETEMP' order by binuppervalue"

BINID       BINUPPERVALUE
----------- --------------------
          1                    1
          2                    1
```

```
           3                     2
           4                     3
           5                     4
           6                     5
           7                     7
           8                    10
           9                    13
          10                    17
          11                    23
          12                    31
          13                    42
          14                    56
          15                    74
          16                   100
          17                   133
          18                   177
          19                   237
          20                   316
          21                   421
          22                   562
          23                   749
          24                  1000
          25                  1333
          26                  1778
          27                  2371
          28                  3162
          29                  4216
          30                  5623
          31                  7498
          32                 10000
          33                 13335
          34                 17782
          35                 23713
          36                 31622
          37                 42169
          38                 56234
          39                 74989
          40                100000

  40 record(s) selected.
```

There are six types of WLM histograms:

- 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 a service subclass or for a work action applied to the work class.

- Coordinator activity execution time (CoordActExecTime)

  This is a histogram of the time that non-nested activities spend executing at the coordinator partition. Execution time does not include time spent initializing or queued. For cursors, execution time includes only the time spent on open, fetch, and close requests.

  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 that 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 request execution times. It includes all requests on both coordinator and non-coordinator partitions, including those requests not associated with an activity.

  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 will 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

The histogram template defines the range. The default template is SYSDEFAULTHISTOGRAM.

Example 4-42 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 the ALTER HISTOGRAM TEMPLATE statement or the CREATE HISTOGRAM TEMPLATE statement to create your histogram template.

Example 4-42 The range of SYSDEFAULTHISTOGRAM

```
>db2 "SELECT binid,binuppervalue FROM syscat.histogramtemplatebins WHERE
templatename='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.
```

To use histograms for a service class, follow these steps.

1. Create and activate an event monitor.

   Use the CREATE EVENT MONITOR statement to create a statistics event monitor. Example 4-43 shows how to create a statistics event monitor that writes data to a table.

   Example 4-43 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), 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 4-44 shows how to set the event monitor active.

   Example 4-44 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 4-45 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 4-45 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 4-46 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 4-46 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, the WLM_COLLECT_INT database configuration parameter sets 0.

5. Look at the statistics.

   Example 4-47 shows how many rows are stored in the HISTOGRAMBIN_DB2STATISTICS table. The table row count can be used for estimate the sizing disk space. 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 4-47 Counting the HISTOGRAMBIN_DB2STATISTICS table

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

COUNT
-----------
        246

  1 record(s) selected.
```

Example 4-48 shows the CoordActExecTime histogram for SYSDEFAULTUSERCLASS. The `BIN_TOP` value `-1` means infinity. If any activity runs over `21600000` milliseconds, the `NUMBER_IN_BIN` for the infinite bin would not be 0. Six activities had an execution time between `0` and `1` milliseconds.

Example 4-48 Analyses of the CoordActExecTime for SYSDEFAULTUSERCLASS

```
>db2 "SELECT bin_top, number_in_bin FROM histograms WHERE
histogram_type='CoordActExecTime' and
service_superclass='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 4-49 shows the CoordActLifetime histogram for SYSDEFAULTUSERCLASS. Five activities are counted in the lowest bin. One activity has the longest lifetime between `3046` and `4647` milliseconds.

Example 4-49 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 4-50 shows the CoordActQueueTime histogram for SYSDEFAULTUSERCLASS. All 11 activities are counted in the lowest bin because nothing is queued in this example.

Example 4-50 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

WLM sample scenarios - mixed OLTP and DSS environment

In many customer environments, combining both Online Transaction Processing (OLTP) and Decision Support Systems (DSS) into a single environment is needed for more timely information and for maintaining a competitive advantage. One of the biggest challenges in a mixed OLTP and DSS environment is developing a proactive setup that allows coexistence without sacrificing throughput of the OLTP environment to the larger DDS queries.

We discuss the following topics in this chapter:

- The setup environment, which illustrates how to identify, manage, and monitor such a mixed environment by using WLM
- Using TCP-H benchmark data to build the environment
- Simple queries that simulate OLTP workloads

5.1 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.
- Proactive management of the availability of critical OLTP resources to achieve a consistent OLTP transaction response time.

In this chapter we demonstrate how to use WLM to achieve these objectives. Following are our business objectives for this exercise:

- 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 need than the analysis reports.
- All other workloads, such as administrative tasks, are expected to only be about 50 - 100 tasks a day.
- Database backups are part of the administrative workload. Several long-running tasks are performed, such as tablespace backups. These are not supposed to run during the prime time shift (8 am - 8 pm) and are not to interfere with the OLTP or DSS workloads.

5.2 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 do not run into the prime shift.
- OLTP
  - Prime time shift is 8:00 am to 8:00 pm.

  - Runs as executable oltp.exe and runs as highest priority.
  - Report the average response times to make sure they 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
  - Run as executable dss.exe and run as lowest priority. If necessary, queue reports to limit their impact on OLTP and DSS queries.

We are now ready to complete our worksheet for identifying our workloads; see Table 5-1.

Table 5-1 Workload worksheet - identifying our workloads

Task	Business requirements	Identification	Action
Admin	Manages the database environment	group ID = 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 user ID = 'BATCH' and utility LOAD	Report the times, duration, and frequency of tasks
OLTP	Prime time shift is 8 am - 8 pm. Queries need to complete in < 1 second	executable = oltp.exe	Assign highest priority, report average response times
DSS	Prime time 8 am to 8 pm	Ad hoc queries and reports run under dss.exe	Identify ad hoc separately from reports
_ DSS queries	Must complete 90% < 5 minutes	group ID = DSSGROUP	Limit impact on OLTP Report average response times
_ DSS Analysis Reports	Must complete all reports daily	exec = dss.exe	Limit impact on OLTP and ad hoc queries

5.3 Manage the work

Given the worksheet details, we are now ready to build our configuration. Our build process is done in two steps:

1. We verify that our configuration is performing the tasks we want.
2. Then we can adjust the configuration to queue or stop execution on the workloads that are impacting our business objectives.

Using our basic monitoring setup in Chapter 4, we can add the additional requirements to our configuration, as shown in Example 5-1.

Example 5-1 Mixed workload configuration

```
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;
  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;

CREATE WORKLOAD WL_OLTP APPLNAME  ('oltp.exe') DISABLE SERVICE CLASS OLTP UNDER
HIGHLVL POSITION AT 1;

CREATE WORKLOAD WL_BATCH CURRENT CLIENT_USERID ('BATCH') DISABLE SERVICE CLASS
BATCH UNDER HIGHLVL POSITION AT 2;

CREATE WORKLOAD WL_PROD_RPT APPLNAME  ('dss.exe') DISABLE SERVICE CLASS
PROD_RPT UNDER HIGHLVL POSITION AT 3;

CREATE WORKLOAD WL_PROD_QRY SESSION_USER GROUP ('DSSGROUP') DISABLE SERVICE
CLASS PROD_QRY UNDER HIGHLVL POSITION AT 4;
CREATE WORKLOAD WL_ADMIN SESSION_USER GROUP ('DB2ADM') DISABLE SERVICE CLASS
ADMINS UNDER HIGHLVL POSITION AT 5;
```

5.3.1 Enabling the instance user ID to alter AIX priorities

The normal user rights given when creating an AIX user ID by default do not give the needed authority to allow the instance owner to adjust agent priorities. This capability must be added before any agent priorities can be changed by DB2. Two additional capabilities are needed. The following command adds the needed capabilities:

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE db2inst1
```

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 5-2

Example 5-2 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=
```

5.3.2 Creating the service classes definitions

Our first service class is our super class, HIGHLVL. Under that, we can create all our subclasses and indicate what level of information we want to capture.

- ADMINS

  Because we do not 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 the SCSTATS table and the HISTOGRAMBIN tables.

- PROD_RPT

  The reports 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 the activity levels. The second collection is added using the ALTER SERVICE CLASS.

  These two levels provide 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

  These transactions are very short-lived and need to run at the highest priority in order to insure the business requirement of running in under 1 second can be maintained. The same data collection levels are used as for PROD_RPT and PROD_QRY.

5.3.3 Creating the workload definitions

Our setup is similar to Chapter 3, “Customizing the WLM execution environments” on page 49, so we only highlight the additional information here.

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.

Note: The workload definitions are searched in the order of their position. To minimize search times, put the most critical workloads at the lowest position number.

5.3.4 Finalizing the setup

Lastly, we grant the use to our workloads and enable the service classes and workloads as shown in Example 5-3.

Example 5-3 Granting WLM permissions and enabling the service classes and 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;
GRANT USAGE ON WORKLOAD WL_OLTP TO PUBLIC;

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;

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;
```

5.4 Monitoring the work

Having implemented our workload management strategy, we collect and monitor the data. The data from the SCSTATS_BASIC_MON and HISTOGRAMBIN_BASIC_MON tables are the tables we are using to demonstrate what monitoring data is used to measure the effectiveness of our WLM strategy. These two tables give a useful 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.

5.4.1 Checking 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. We used the following command:

```
db2pd -db wlmdb -serviceclasses
```

Example 5-4 displays the service class definitions and shows that the OLTP already is being used (see the highlighted lines).

Example 5-4 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, we run the following command:

```
db2pd -db wlmdb -agent
```

Example 5-5 shows that we are running eight OLTP transactions and they are all running at a priority of -10.

> **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).

Example 5-5 db2pd agent priorities

```
->db2pd -agent -dbpartition 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] AgentEDUID Priority   Type     State
ClientPid  Userid   ClientNm Rowsread   Rowswrtn   LkTmOt DBName
0x078000000023ACC0 67157    [001-01621] 9879       0          Coord
Inst-Active 2728214    dssuser  db2stmm  0          0          NotSet WLMDB
0x078000000023EAE0 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
....
```

To check the prefetchers, we first switch to a data partition then run the **db2pd** command as follows:

```
export DB2NODE=1
db2 terminate
db2pd -db wlmdb -edu -dbp 1
```

Example 5-6 displays the output.

Example 5-6 Output from db2pd -edu for prefetchers

```
->db2pd -db wlmdb -edu -dbp1 | egrep "EDU Name|db2pfchr"
EDU ID    TID           Kernel TID     EDU Name                              USR          SYS
9658            2941051       5906937           db2pfchr (WLMDB) 1                   0.001285
0.000531
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
5803      5803          2044119        db2pfchr (WLMDB) 1                    0.094211     0.057868
```

5.4.2 Monitoring and analyzing the service classes

To begin with, we get a high level view of our workloads by looking at the HISTOGRAMBIN_BASIC_MON table. Using a query, we can see the histogram data. Using a query as shown in Example 5-7, we can plot graphs that give us an overview of how our system is performing.

Example 5-7 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 ;
```

The first chart 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 5-1 shows the request execution time chart.

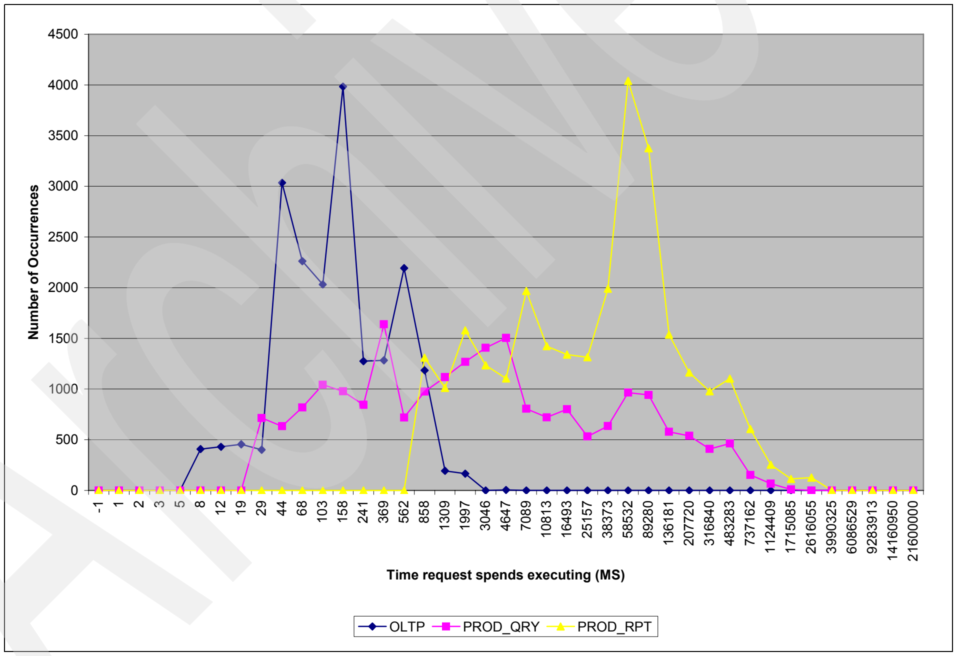


Figure 5-1 Request execute time

Using the histogram, we can create a graph of the number of queries by request execution times. This give us a quick glance at the types of queries (service class) and how many queries ran within each time slot. Our OLTP is running between .008 and 4.647 seconds. Our objective is to run all of our OLTP transactions under 1 second, but we can see that 346 queries are outside of our business objective.

Using the request execution times gives a useful picture because the request time is the total time for the query, including any wait times. This objective will need further analysis in order to determine why this is occurring: is it a problem with the OLTP queries themselves, or some outside influence that is causing them to not meet the runtime objectives?

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, with approximately 52% of these queries completing in under 58 seconds. The longest-running reports run in 2616.055 seconds or 43.6 minutes. This appears to be a significant number of reports but with only a limited amount of data, it is difficult to draw any meaningful conclusions. All were completed, but did they interfere with any of our objectives? To determine this, we need more information.

Next, we turn our attention to the arrival rates of our workloads. Figure 5-2 illustrates the arrival rates.

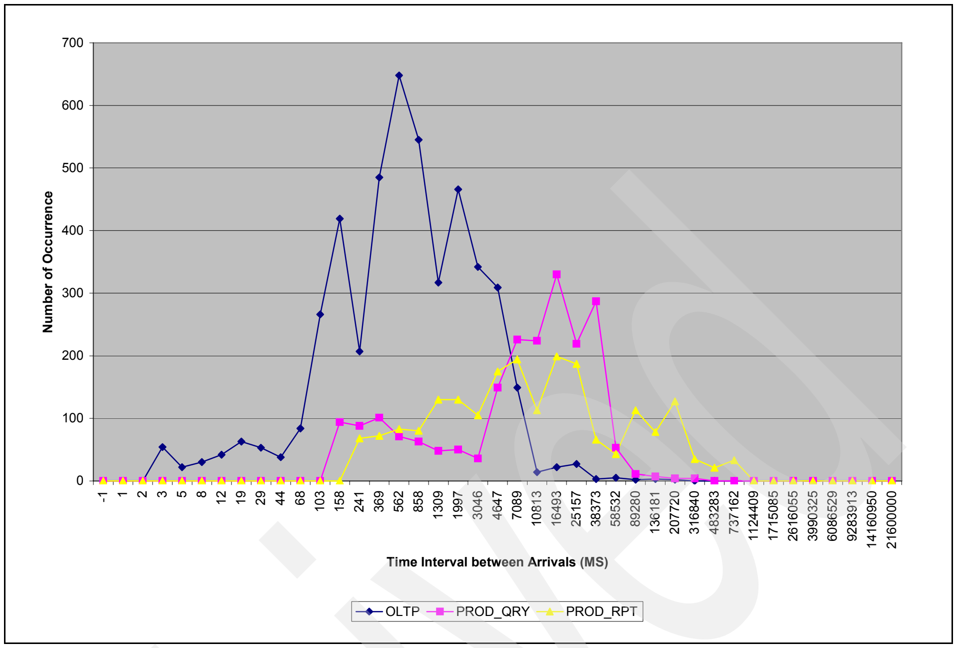


Figure 5-2 Arrival rates

Here we see how "fast" workloads are arriving in our system. So, are we overloading our system at critical times? That is the question we want to answer. Using the coordinator activity arrival times, we can chart how often workloads are hitting the system.

Due to the large number of reports shown on the previous chart, and because we missed our objective for some of the OLTP queries, our focus is on the Production reports (PROD_RPT). From this graph we can see the PROD_RPTS peak at nearly 200 arrivals that are 25 seconds apart. Is this the expected arrival rate? Does the high arrival rate of PROD_RPTS interfere with our OLTP transactions? The peak arrival rate for OLTP queries is 648 queries arriving 5.6 seconds apart. We need to examine when the PROD_RPT queries arrive by looking at the aggregate service class statistics (SCSTATS_BASIC_MON) table.

From Example 5-1 on page 126, we can focus in on the request execution times for specific time periods. Here we can see when the request execution times for OLTP missed 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; see Figure 5-3.

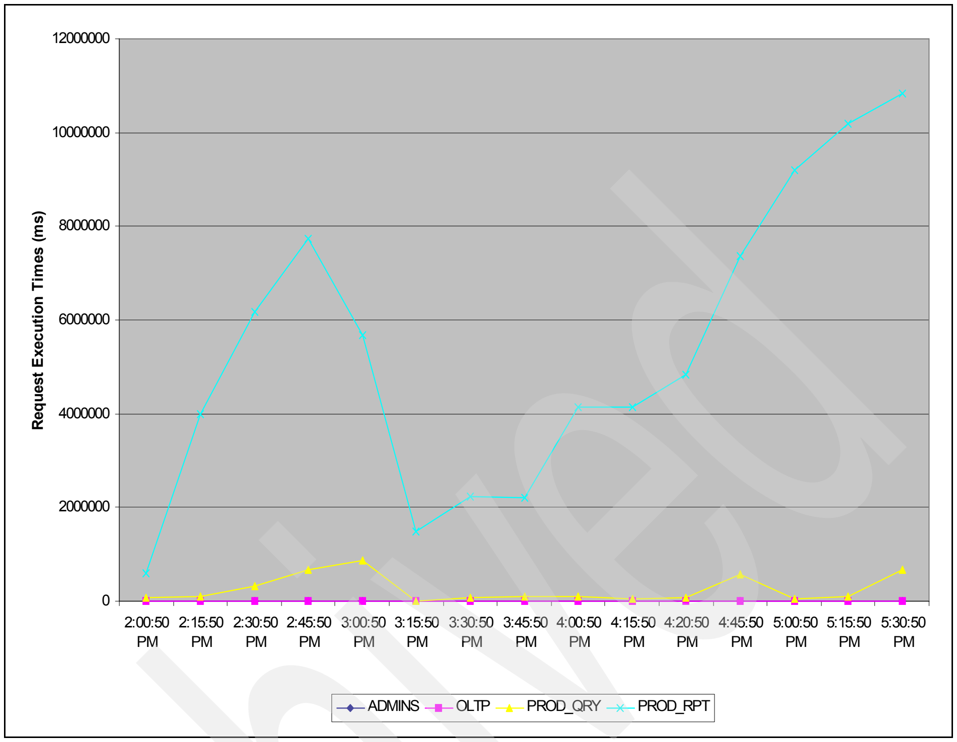


Figure 5-3 Missing target

Next, we need to look for the cause of the excessive OLTP request execution times. The queries could be running long for several reasons, including that the queries need tuning or that machine resources could be in short supply at various times.

We can continue our analysis by looking at how much execution time is being used by the various types of queries. Request execution time can be used for such analysis, but with caution. The request execution time is the total time it takes for a query to complete. This includes wait times as well as CPU time. Nonetheless, we can use request execution times as a preliminary analysis tool.

From the chart shown in Figure 5-3, we have a clue. The PROD_RPT total request executions times are also quite long, especially during the middle of the day when many OLTP queries might also running. Because the OLTP queries are quite short, they do not show up well on this chart, but our focus is on the PROD_RPT request execution times. Because this represents a total request execution time for all queries within each service class, we cannot tell if the long execution time is the result of a few queries or many queries. So, what else can we analyze to give a clearer picture?

In Figure 5-4, we look at the arrival times for workloads during the same time frame as the Figure 5-3 chart on request execution times. Using the concurrent activity top, we can create a chart to show when the highest concurrency of activities for each service class is taking place.

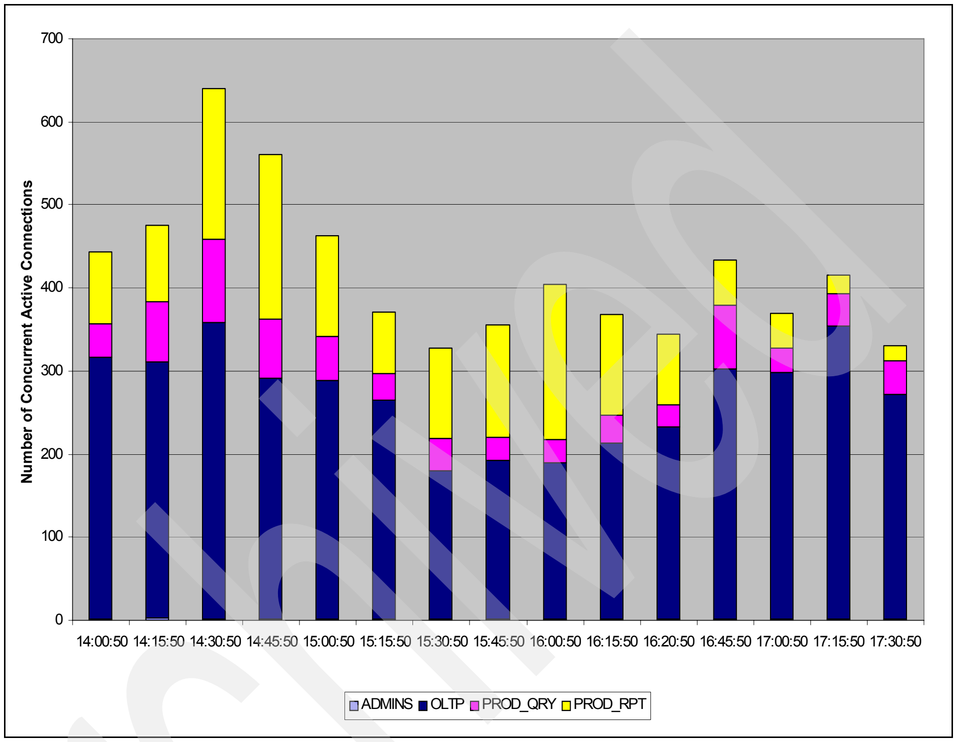
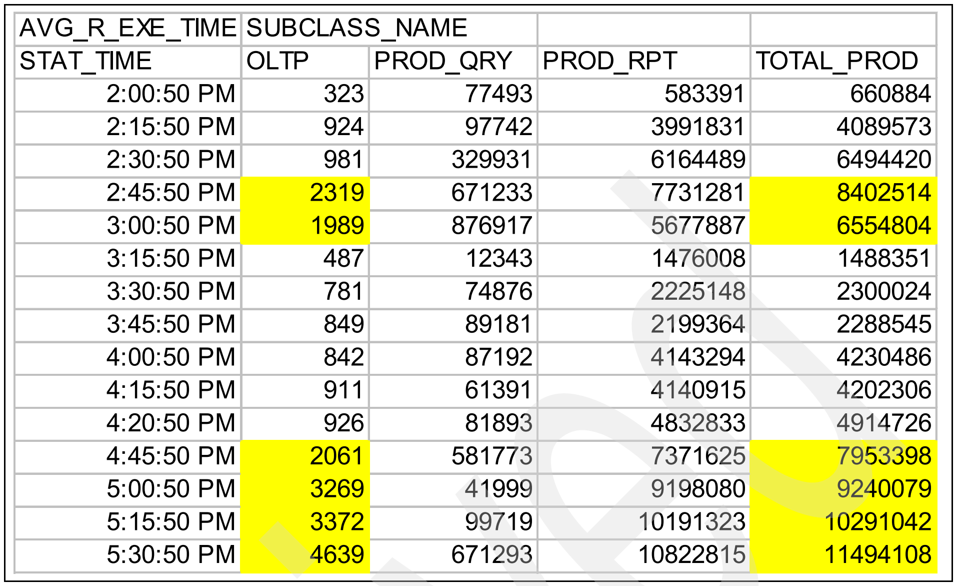


Figure 5-4 Arrival time

Here we see many workloads arriving at the same time by viewing the concurrent active coordinator active. Notice the larger number of PROD_RPT workloads.

Taking a closer look at the request execution times, we get a clearer picture of the relationship between OLTP and PROD_RPTS. Using the data from the SCSTATS table, we can construct a spreadsheet as shown in Figure 5-5 on page 140. (Note that ADMIN subclass has been omitted for simplicity.)



AVG_R_EXE_TIME	SUBCLASS_NAME			
STAT_TIME	OLTP	PROD_QRY	PROD_RPT	TOTAL_PROD
2:00:50 PM	323	77493	583391	660884
2:15:50 PM	924	97742	3991831	4089573
2:30:50 PM	981	329931	6164489	6494420
2:45:50 PM	2319	671233	7731281	8402514
3:00:50 PM	1989	876917	5677887	6554804
3:15:50 PM	487	12343	1476008	1488351
3:30:50 PM	781	74876	2225148	2300024
3:45:50 PM	849	89181	2199364	2288545
4:00:50 PM	842	87192	4143294	4230486
4:15:50 PM	911	61391	4140915	4202306
4:20:50 PM	926	81893	4832833	4914726
4:45:50 PM	2061	581773	7371625	7953398
5:00:50 PM	3269	41999	9198080	9240079
5:15:50 PM	3372	99719	10191323	10291042
5:30:50 PM	4639	671293	10822815	11494108

Figure 5-5 SCSTATS table of Request Execution Times

From this, we see that the OLTP queries are over our objective of one second whenever the total request execution times are greater than 6500 seconds. We also see some of the longest-running PROD_RPT queries during this time.

Now, we want to learn more about the PROD_RPTS for this time frame. In Figure 5-6 on page 141, we focus on the costs of these reports.

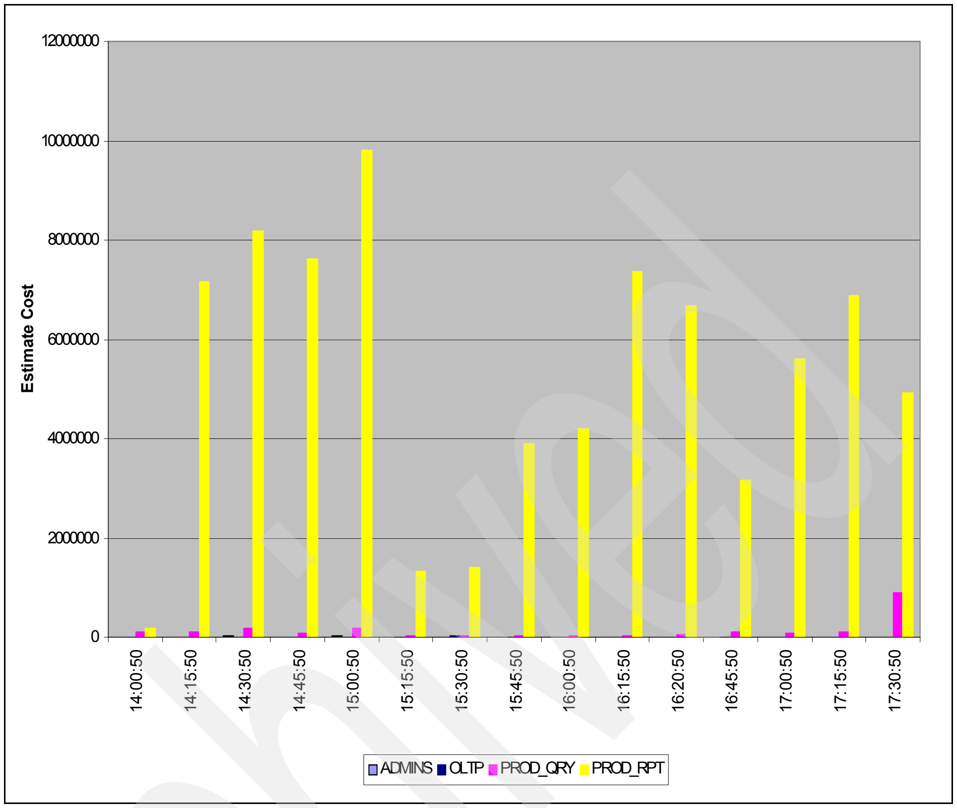


Figure 5-6 Report cost

Here we see the PROD_RPT queries cost vary greatly, but the more expensive reports are being run during the same time frame as when our OLTP queries begin to miss our business objectives. The PROD_RPT queries cost is greater than 4,000,000 during our critical times for OLTP. We will want to limit them during this time to lessen the impact on our OLTP queries.

5.5 Summary

Looking back at our earlier histogram chart, we see that the PROD_RPTS are running near the top of the costs reported in the histograms. To summarize our analysis so far, we see many 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.

We want to take a global approach because PROD_RPTS costs are much higher than any other “known” workload. Based on the information we have collected and analyzed in this chapter, we can develop our next workload management strategy.

In Figure 5-6 on page 141, we see that we need to limit expensive PROD_RPT queries. These have been identified as queries with a cost greater than 4,000,000 timerons. Additionally, we want to impose this limit based on the total number of concurrent coordinators, so that during off-peak times, these queries can run unlimited.

In Figure 5-4 on page 139, we see that during our slow OLTP times, we are running in excess of 400 concurrent coordinators. To identify this new WLM requirement we develop a management strategy, as shown in Example 5-8.

Example 5-8 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 4000000 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 ;
```

This additional WLM strategy states that when the number of concurrent database coordinators is > 400, we want to queue any workloads that have a timeron cost >= 4,000,000. 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.

6

AIX Workload Manager considerations

There are two different kinds of workload management solutions available: operating system (OS)-level workload management solutions, and application-level workload management solutions. DB2 Workload Manager (DB2 WLM) is an example of an application-level workload management solution. AIX Workload Manager (AIX WLM) is an example of an operating system-level workload management system.

DB2 9.5 Workload Manager supports operating system WLM on AIX only. Integrating DB2 WLM with AIX WLM provides you with even more capability to manage and control the workloads and resources on your database system.

In this chapter we discuss the following topics:

- AIX WLM overview
- Using DB2 WLM and AIX WLM

6.1 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 to be the most important application running on the server. AIX WLM is an ideal solution for protecting database applications from being interfered with by the other applications running on the same server.

For detailed information about AIX WLM, refer to the IBM Redbooks publication *AIX 5L Workload Manager (WLM)*, SG24-5977, which is available at the following site:

http://www.redbooks.ibm.com/abstracts/sg245977.html

6.1.1 Service classes

Conceptually, AIX WLM is similar to DB2 WLM. AIX WLM has workloads and two-level hierarchical service classes. In addition to the predefined service classes, AIX WLM also has user-defined superclasses.

Each superclass can have both user-defined and predefined subclasses. And no process can belong to a superclass without also belonging to a subclass.

Every process in the system is mapped to the predefined default subclass, unless it is explicitly defined to another subclass by the administrator.

Predefined superclasses

The predefined AIX WLM superclasses are as follows:

- Default superclass

  All user processes belong to the default superclass, unless they are explicitly defined to map to another superclass.

- System superclass

  The system superclass is the default superclass for all privileged processes owned by root.

- Shared superclass

  The shared superclass receives all memory pages that are shared by processes that 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 that are unmanaged by AIX WLM. This superclass does not have any shares or limits for any resources. This superclass does not have any subclasses. No processes will be assigned to this superclass.

Figure 6-1 illustrates how resources are assigned with a default AIX WLM configuration.

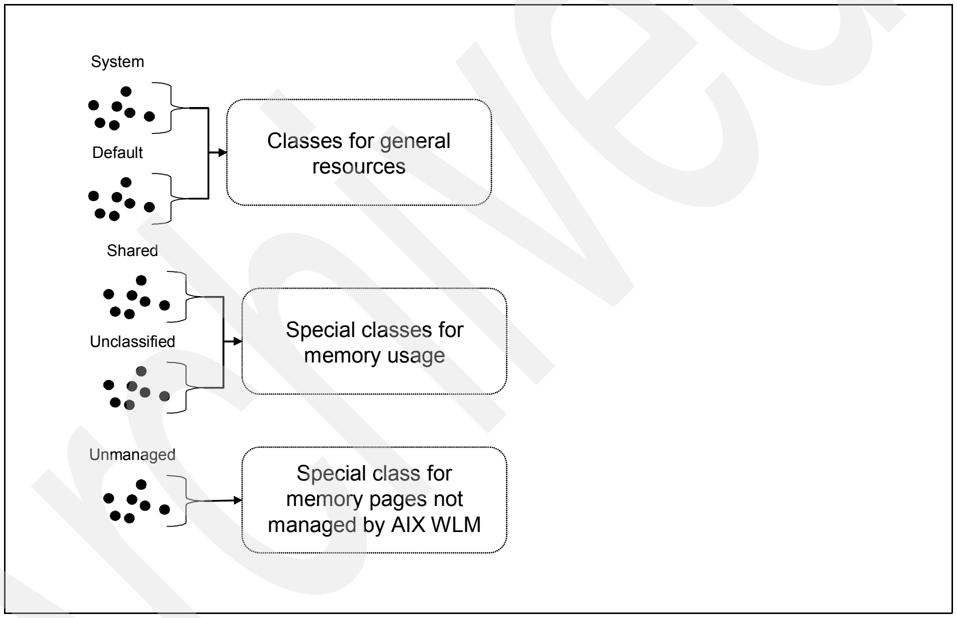

System
Default
Classes for general resources
Shared
Unclassified
Special classes for memory usage
Unmanaged
Special class for memory pages not managed by AIX WLM

Figure 6-1 AIX WLM default service classes

Predefined subclasses

Subclasses can have only one superclass. There are two predefined subclasses, as explained here:

- Default subclass

  All processes which are not defined to any other subclass will be assigned to the Default subclass.

- Shared subclass

  Like the Shared superclass, the Shared subclass contains all the memory pages that contain processes belonging to more than one subclass under the same superclass.

Tier

The tier defines the importance of the 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 tier level 0 classes take up all the system resources, then tier level 1 classes will not get any resources. If 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: This can be up to 16 characters in length.
- Tier: This has values from 0 to 9. The tier value enables you to prioritize groups of classes. The default and highest priority value is zero (0).
- Inheritance: This defines whether a child process or thread inherits the class assignment of its parent.
- Authuser and authgroup: This is used give users or groups the right to manually assign processes to a certain class. (These attributes can be used *only* for superclasses.)
- Adminuser and admingroup: This is used to delegate administration rights to certain user IDs or groups. (These attributes can be used *only* for superclasses.)
- Resource set: This limits the processes in a given class to a subset of the system's physical memory and processors. A valid resource set is composed of memory and at least one processor.
- Localsmh: This 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

A share defines the proportion of a resource that the process in a service class can get. With shares, you can allocate a certain amount of CPU, memory, or I/O resource to each service class. The resources will be allocated to a service class in a relative way, depending on the number of total shares and the number of shares the service class itself has.

For example, if we have a total of 1000 CPU shares for all superclasses, and individual superclass db2Def has 400 CPU shares, then db2Def will get 40% of all CPU resources. If we increase the number of total CPU shares for all superclasses to 2000, then with its 400 CPU shares, db2Def would have 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

  This specifies the minimum percentage of a resource that will always be granted to a service class. The default is zero (0).

- softmax

  This specifies the maximum percentage of the resource that can be assigned to a service class when there is contention, as explained here:

  – If there is no contention, then the service class can obtain more resources than that 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

  This specifies the 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: using manual classification, or using automatic classification. When you want to assign processes manually to certain service classes, you can do it by using the command **`wlmassign <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 that 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 a process or thread will be mapped.
- User and Group: These define by which user ID or group ID a process is owned.

- Application: This defines the full path name of the application, and it can contain wild cards.
- Tag: A label for identifying processes and threads that will be assigned by the 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 a special rules file. /etc/wlm/current/rules is the rules file for all superclases. For subclasses, rules files are located under the corresponding superclass directory. All superclass directories for running WLM configuration can be found under directory /etc/wlm/current/<superclass name>.

6.1.2 Monitoring

There are many AIX WLM-aware operating system monitoring tools to monitor AIX WLM:

- **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>**.

6.1.3 Configuring AIX WLM

All AIX WLM-related configuration files are located in the /etc/wlm directory. The subdirectories contain different configuration sets, as shown in Example 6-1.

Example 6-1 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 the new configuration set as the current configuration:

   ```
   wlmcntrl -d testenv
   ```

3. Create superclass and subclasses.

   AIX WLM can be configured using smitty or by using commands and editing WLM-specific configuration files. Example 6-2 shows commands to create superclasses and subclasses.

   Example 6-2 Creating AIX WLM service classes - sample 1

   ```
   mkclass -a inheritance=no -c shares=60 SaleSupClass
   mkclass -a inheritance=no -c shares=35 SaleSupClass.App1
   mkclass -a inheritance=no -c shares=25 SaleSupClass.App2
   mkclass -a inheritance=no -c shares=40 MktSupClass
   ```

   In Example 6-3, we created only one 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.

   If you only have one superclass in the system, by default it obtains all the resource shares (100%). You can set the share limit by giving a different share amount.

   Example 6-3 Creating AIX WLM service classes - sample 2

   ```
   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 the specific rules file. If you prefer to use smitty, use the command **`smitty wlm`** to start it, then 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 management requirement that we want to achieve.

   – app1_batch:
     The binaries are in /opt/App1/batch/bin. app1_batch collects the

performance statistics of app1_app. app1_batch is not as critical as app1_app and can have less of a share of CPU time.

We defined our control rules by editing /etc/wlm/testenv/rules, as shown in Example 6-4. We mapped both applications, in directories /opt/App1/bin and /opt/App1/batch/bin, to superclass HighLevel.

Example 6-4 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 edited this file as shown in Example 6-5.

   Example 6-5 Subclass mapping

   ```
   * class resvd user group application type tag
   Prod    -       -       -       /opt/App1/bin/* -       -
   Utils   -       -       -       /opt/App1/batch/bin/*   -       -
   ```

6. Update configuration settings.

   After 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 HighLevl.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, for a total of 100 shares.

Figure 6-2 shows that our system reflects this setting. The x axis represents time and the y axis represents the percentage of CPU that is consumed. The data is from the nmon monitor tool. (We discuss this tool in “nmon” on page 167.)

In the figure, we see that by setting shares 60/40, subclass HighLevel.Prod gets priority 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.

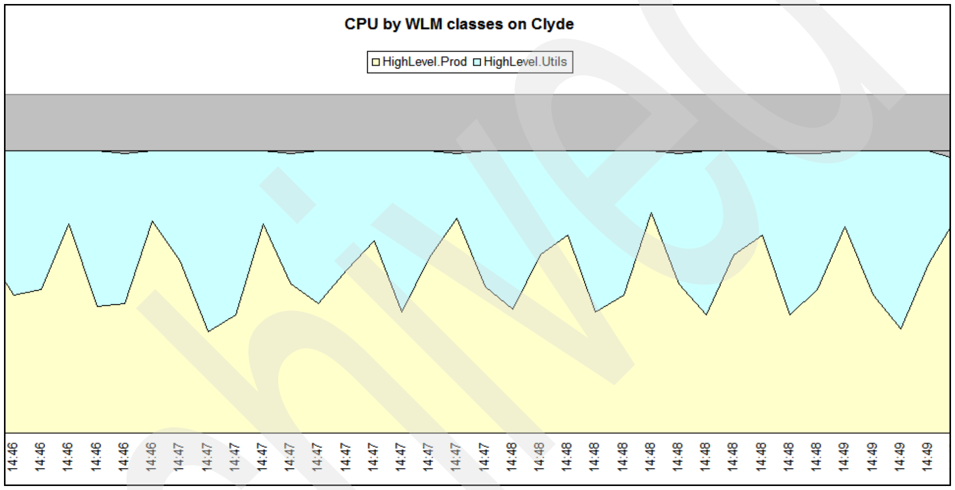


Figure 6-2 CPU by service class

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 at average data in the same time period, we find that the average CPU for HighLevel. Prod was 60%, and the average CPU for HighLevel.Utils was 40%, as illustrated in Figure 6-3.

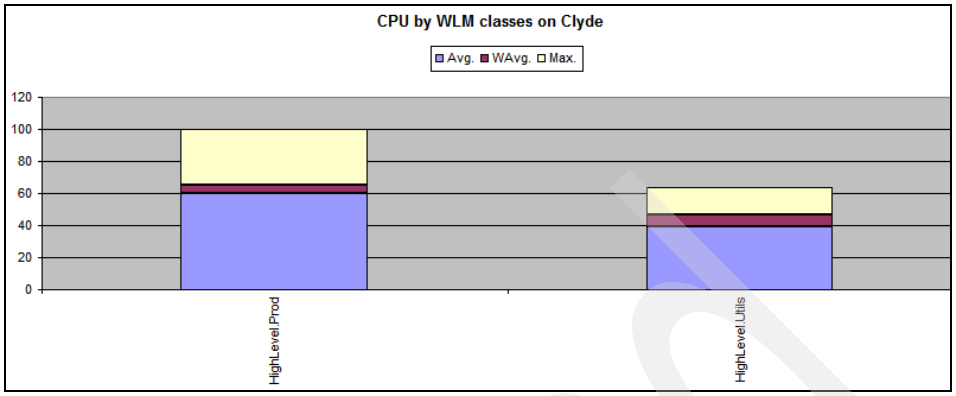


Figure 6-3 Average CPU time by service class

We ran the same test without defining AIX WLM CPU shares. As you can see from Figure 6-4, the result looks much different than when CPU shares were defined.

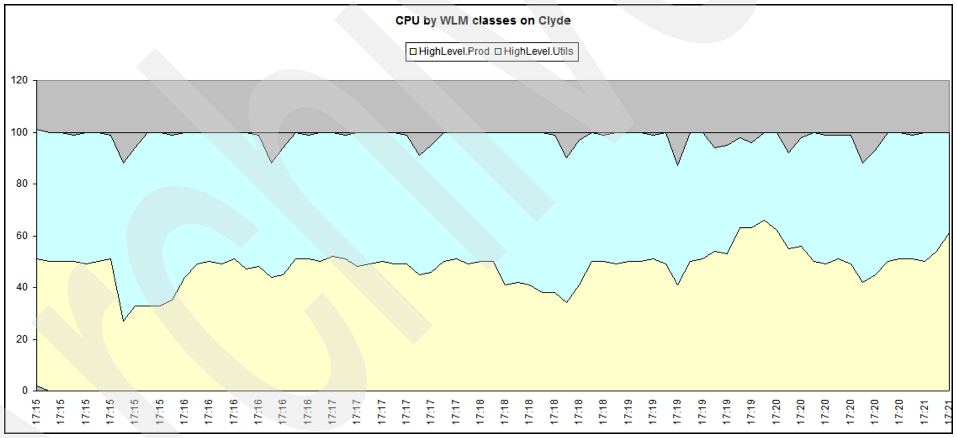


Figure 6-4 CPU by service class without CPU shares

There is a major difference between average values as well, as shown in Figure 6-5 on page 153. On average, there was no difference on consumed CPU time between the applications. In fact, there was no guarantee that app1_app could get the resources it needed. This could have led to poor service quality and a violation of the business objective.

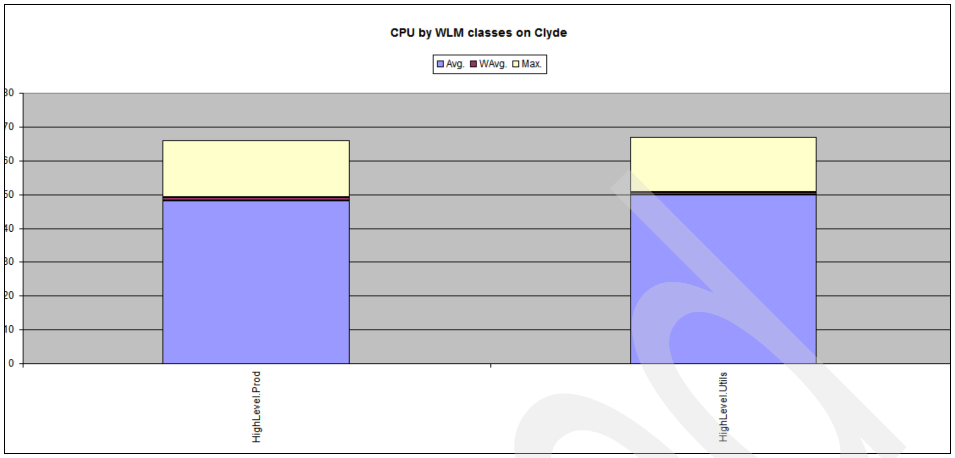


Figure 6-5 Average CPU time by service class without CPU shares

6.2 Using DB2 WLM and AIX WLM

In this section we discuss how to integrate DB2 WLM with AIX WLM. We introduce general guidelines for planning and designing an integrated environment. We also provide useful examples and discuss monitoring tools.

6.2.1 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 the prefetch priority on the DB2 WLM service class. You can set prefetch priority in DB2 WLM and define CPU prioritization through AIX WLM, simultaneously.

At the time of writing, AIX WLM does not provide I/O priority at the thread level. When you integrate DB2 WLM with AIX WLM, you can only set AGENT PRIORITY for DB2 service classes to “Default”, because it will prevent setting an outbound correlator to be used with the operating system WLM.

AIX WLM is capable of using application tags to map processes and threads to desired service classes. You define application tags through the DB2 WLM configuration by setting an outbound correlator for your service classes. After integrating your WLM environments, AIX can manage the CPU utilization for DB2 service classes.

6.2.2 Mapping schemes

Before you begin using DB2 WLM and AIX WLM, you need to decide how to map different DB2 service classes with AIX WLM service classes. From the conceptual management point of view, the database system has two hierarchies:

- Instance
- Database

When adding the two DB2 WLM hierarchies (superclasses and subclasses), there is a total of four levels of hierarchy. If AIX WLM, which has a two-level hierarchy, is going to be used to manage the database system from the instance level, then you have to carefully design the mapping between the DB2 instance, database, service classes, and AIX services classes.

You can choose from two different mapping schemes: the flat mapping scheme, and the 1:1 mapping scheme.

Figure 6-6 illustrates the flat mapping scheme. This scheme is useful on servers that are used not only as the DB2 database server, but also have other applications running. In such a 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 the AIX WLM DB2 superclass.

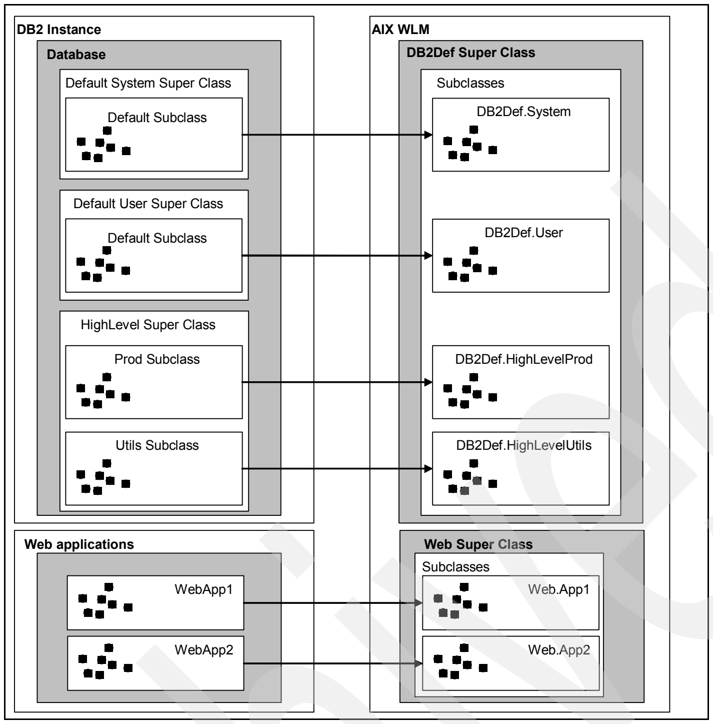


Figure 6-6 Flat mapping scheme

When you are running a dedicated database server, we recommend that you use the 1:1 mapping scheme, as illustrated in Figure 6-7.

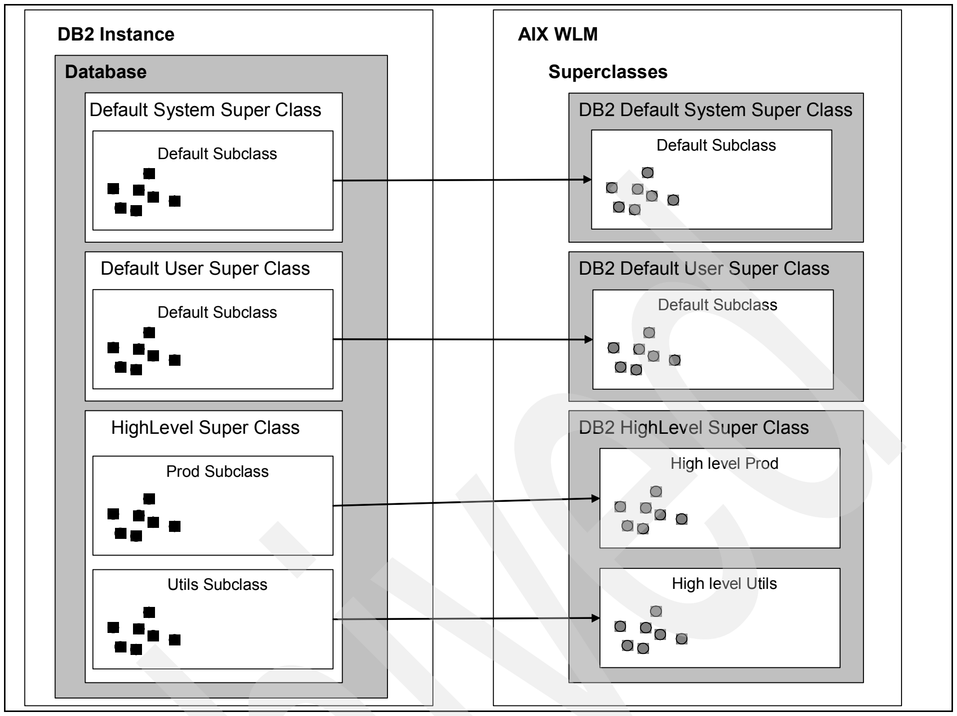


Figure 6-7 1:1 mapping

The main difference between flat mapping and 1:1 mapping is that with 1:1 mapping, you can have identical superclasses and subclasses on both DB2 WLM and AIX WLM. This makes implementation and maintenance much easier. We use a 1:1 mapping scheme for our example in this chapter.

6.2.3 Integrating DB2 service classes with AIX service classes

When integrating DB2 WLM with AIX WLM, careful planning and proper design are the keys to success. You should be able to maintain and monitor your environment easily. If you have already implemented DB2 WLM, determine whether your DB2 WLM configuration can be integrated with AIX WLM. You also have to decide whether to use 1:1 mapping or a flat mapping scheme.

Designing DB2 WLM and AIX WLM integration

Because we are running a dedicated database server, we use 1:1 mapping as the mapping scheme. Figure 6-8 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.

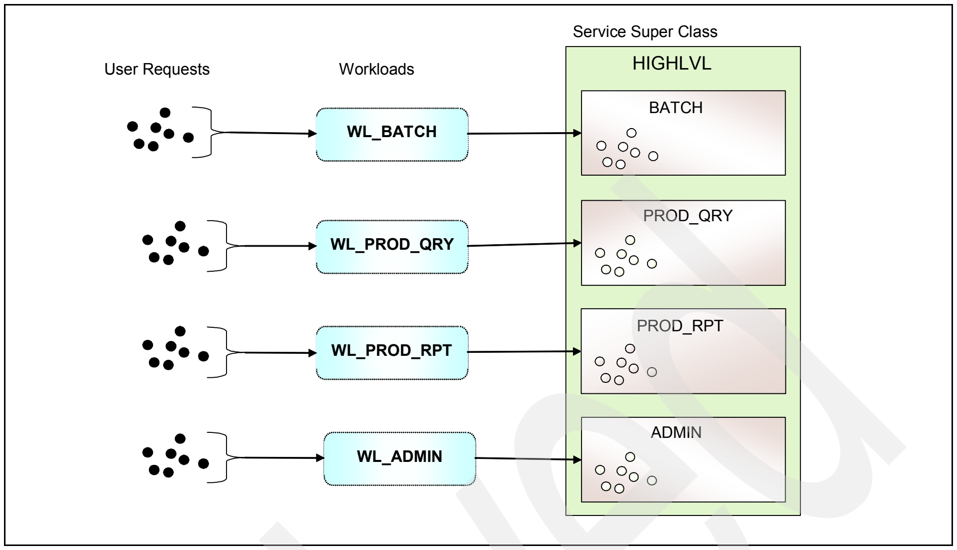


Figure 6-8 User workloads and service classes

In addition to 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 have one default subclass. So, all together we have 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 → db2DefUsr
- SYSDEFAULTMAINTENANCECLASS → db2DefMnt
- HIGHLVL → 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 many shares the service class has relative to the total number of shares. We have to decide how many CPU shares we are going to give our service classes. Figure 6-9 shows the CPU share 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 an under score (_) mark.

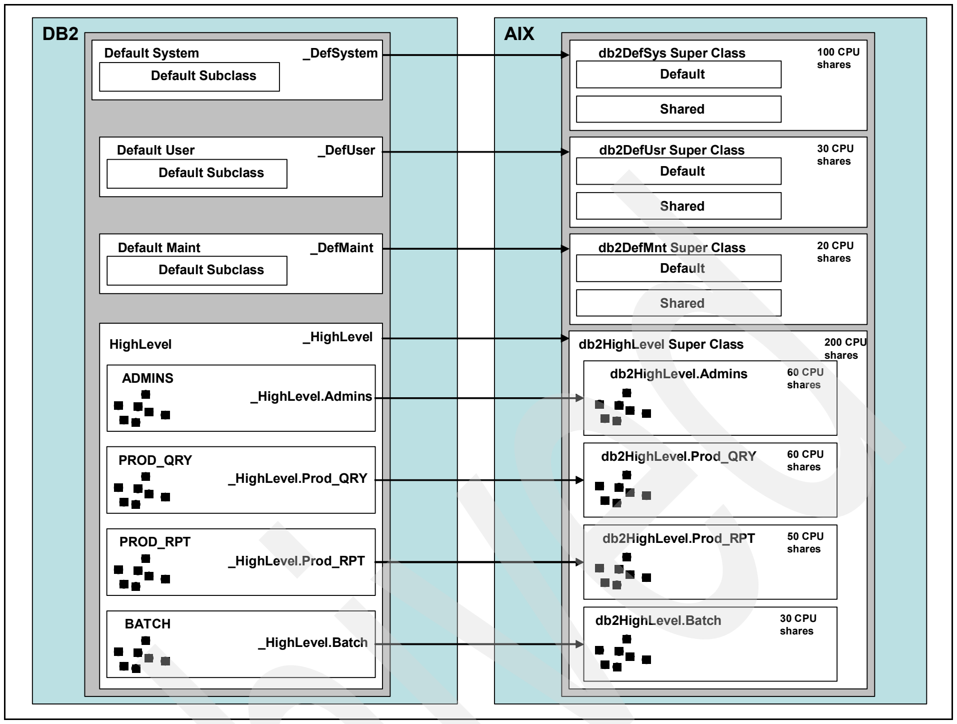


Figure 6-9 1:1 mapping from existing DB2 WLM configuration

Implementing DB2 WLM and AIX WLM integration

Now that we have designed our mapping scheme, we are ready to implement the mapping. We start by creating new configuration set, *production*, for AIX WLM; see Example 6-6.

Example 6-6 Creating new AIX WLM configuration set

```
# cd /etc/wlm
# cp -r template production
# wlmcntrl -ud production
```

Note that using the **wlmcntr -ud** command requires AIX WLM to be already running. If AIX WLM is not running, use the 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 6-7.

Example 6-7 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

Because 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 6-8 shows the contents of /etc/wlm/current/rules. Be aware that entries in the rules file also imply the evaluation order.

Example 6-8 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: Always leave the System class *above* the user-defined classes, so that system processes will go to their default service classes. Also, leave the Default user class *below* your user-defined classes.

Next, we define the mapping rules for subclasses under the db2HighLevel. The corresponding rules file for the db2HighLevel is /etc/wlm/current/db2HighLevel/rules. We edit this file as shown in Example 6-9.

Example 6-9 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 **`wlmcntrl-u`** command to refresh the running AIX WLM configuration so that all the changes will take effect. We check that all our newly-defined service classes are in the system by using the **`lsclass -fr`** command, as shown in Example 6-10.

Example 6-10 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 test our configuration by checking to which AIX WLM service class the processes and threads with AIX WLM tag _HighLevel.Batch are assigned. The **wlmcheck** command can be used to check, as shown in Example 6-11.

Example 6-11 Test mapping rules

```
# wlmcheck -a "- - - - - _HighLevel.Batch"
System
db2HighLevel.Batch
```

As you can see from Example 6-11, 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, then it will be mapped to System Superclass. 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 physical nodes. To set up another node, you only need to copy (using ftp or scp) the AIX WLM configuration directory to that node and activate it. Example 6-12 shows how we set up AIX WLM on the second physical database node Bonnie in our text environment.

Example 6-12 Copying AIX WLM configuration from node to another

```
> scp -r clyde:/etc/wlm/production /etc/wlm/
...
> wlmcntrl -d production
```

Note that the **`wlmcntrl`** command is executed without the **`-u`** flag because the AIX WLM was not running when the command was executed.

Setting up tags for DB2 service classes

After the AIX WLM environment is ready and running, the last step is to "link" DB2 service classes to AIX WLM service classes. This is accomplished by altering the DB2 service classes outbound correlator to associate the AIX WLM tags with them.

Similar to DB2 WLM, every AIX WLM superclass has at least one subclass for processes. On AIX WLM, there are two default subclasses:

- Default: This is used for all processes that are not mapped to any other subclass.
- Shared: All memory pages that contain processes belonging to more than one subclass 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 tasks will be done under the default AIX WLM subclass, which inherits all its attributes from its parent superclass. This is why we do not need to define rules for default AIX WLM subclasses, and also why we do not need to create additional subclasses for default AIX WLM superclasses.

Example 6-13 and Example 6-14 show how to set up an outbound correlator for default and user-defined DB2 WLM service classes.

Example 6-13 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 6-14, we set an outbound correlator for our HIGHLVL service superclass and the subclasses running under it.

Example 6-14 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 shown in Example 6-15.

Example 6-15 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 operating system level mapping

In addition to verifying that the application tags are set properly, we want to be sure that the actual operating system level mapping really happens. We can check whether the default DB2 system service class is getting mapped to the correct AIX WLM service class by using the **ps** command, as shown in Example 6-16.

Example 6-16 Checking that mapping is working for the 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 used to obtain the process ID for the db2sysc process on partition0. Then we used this process ID to find out whether 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 also use the same approach to check whether the default maintenance service class is mapped as defined. Before checking if the user-defined DB2 workloads are mapped to correct AIX service classes, first check whether 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 is for the DB2 WLM user superclass. Figure 6-10 illustrates how CPU resources are shared by AIX WLM service superclasses.

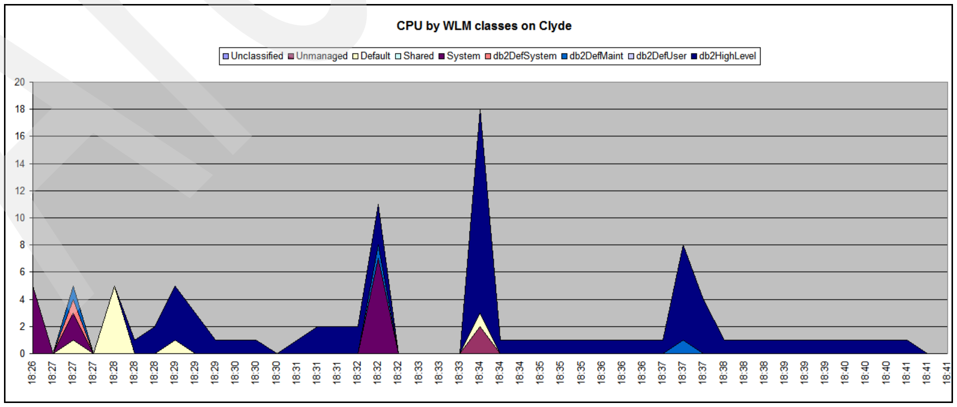


Figure 6-10 CPU by classes

The chart in Figure 6-10 on page 164 shows that our dedicated database server is currently running quite a 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, because currently no workloads are mapped to DB2 WLM service class SYSDEFAULTUSERCLASS. db2HighLevel has some activity, because 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. Because the chart only shows CPU activity, we should also examine memory and disk usage to find out what caused the gap.

6.2.4 Monitoring

AIX offers excellent monitoring tools, and some of them are AIX WLM-aware. These tools provide a quick 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 ever.

Basic AIX WLM monitoring tools

In this section, we concentrate on the following AIX WLM-aware monitoring tools:

- topas
- ps
- wlmstat
- nmon

Note that topas, ps, and wlmstat come with the standard AIX operating system. You can freely download nmon from the following site:

`http://www.ibm.com/collaboration/wiki/display/WikiPtype/nmon`

topas

The topas tool is useful and user-friendly. You can use it to monitor your system in real time. The topas tool provides a good view of overall system performance, and you can include AIX WLM data. You can specify how many top AIX WLM service classes you want to include in topas output by specifying the number of WLM classes using the **`-w`** flag.

Figure 6-11 illustrates the output of **`topas -w 10`**.

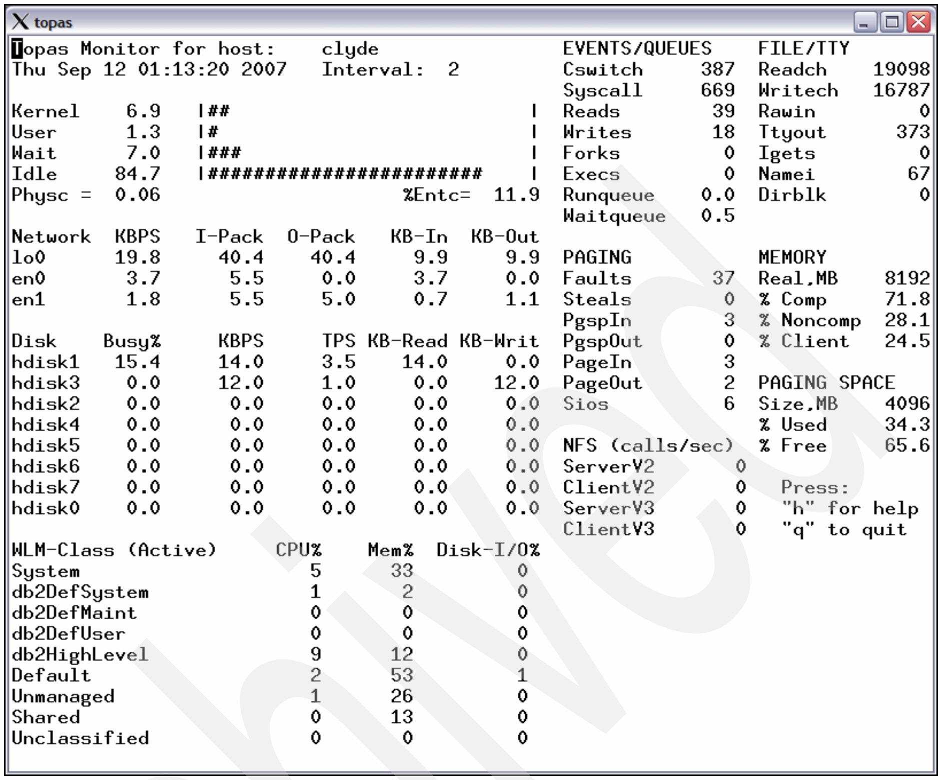

```
Topas Monitor for host:    clyde                          EVENTS/QUEUES    FILE/TTY
Thu Sep 12 01:13:20 2007   Interval:  2                   Cswitch     387  Readch    19098
                                                          Syscall     669  Writech   16787
Kernel    6.9   |##                                  |    Reads        39  Rawin         0
User      1.3   |#                                   |    Writes       18  Ttyout      373
Wait      7.0   |###                                 |    Forks         0  Igets         0
Idle     84.7   |########################            |    Execs         0  Namei        67
Physc =  0.06                        %Entc=  11.9         Runqueue    0.0  Dirblk        0
                                                          Waitqueue   0.5
Network  KBPS   I-Pack  O-Pack   KB-In  KB-Out
lo0      19.8     40.4    40.4     9.9     9.9            PAGING           MEMORY
en0       3.7      5.5     0.0     3.7     0.0            Faults       37  Real,MB    8192
en1       1.8      5.5     5.0     0.7     1.1            Steals        0  % Comp     71.8
                                                          PgspIn        3  % Noncomp  28.1
Disk    Busy%     KBPS     TPS KB-Read KB-Writ            PgspOut       0  % Client   24.5
hdisk1   15.4     14.0     3.5    14.0     0.0            PageIn        3
hdisk3    0.0     12.0     1.0     0.0    12.0            PageOut       2  PAGING SPACE
hdisk2    0.0      0.0     0.0     0.0     0.0            Sios          6  Size,MB    4096
hdisk4    0.0      0.0     0.0     0.0     0.0                             % Used     34.3
hdisk5    0.0      0.0     0.0     0.0     0.0            NFS (calls/sec)  % Free     65.6
hdisk6    0.0      0.0     0.0     0.0     0.0            ServerV2      0
hdisk7    0.0      0.0     0.0     0.0     0.0            ClientV2      0    Press:
hdisk0    0.0      0.0     0.0     0.0     0.0            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 6-11   Using topas for overall system monitoring

To view only real time CPU usage between service classes, use **`topas -W`**. Example 6-17 shows what the output may look like.

Example 6-17   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

The **ps** command is very useful for obtaining information about processes and threads. In Example 6-15 on page 163, we show how to find the mappings between DB2 service classes and AIX WLM service classes from DB2 side by using the **ps** command. For more details about the **ps** command, refer to the AIX manual pages.

wlmstat

If you are familiar with **iostat** and **vmstat**, you should have no difficulty using the **wlmstat** command. It displays all the super classes and service classes with current CPU, memory, and disk I/O usage; see Example 6-18. The **wlmstat** command accepts two numeric input parameters. The first one is interval and the second one is number of cycles. For more detailed usage information about the **wlmstat** command, refer to the **wlmstat** manual pages.

Example 6-18 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

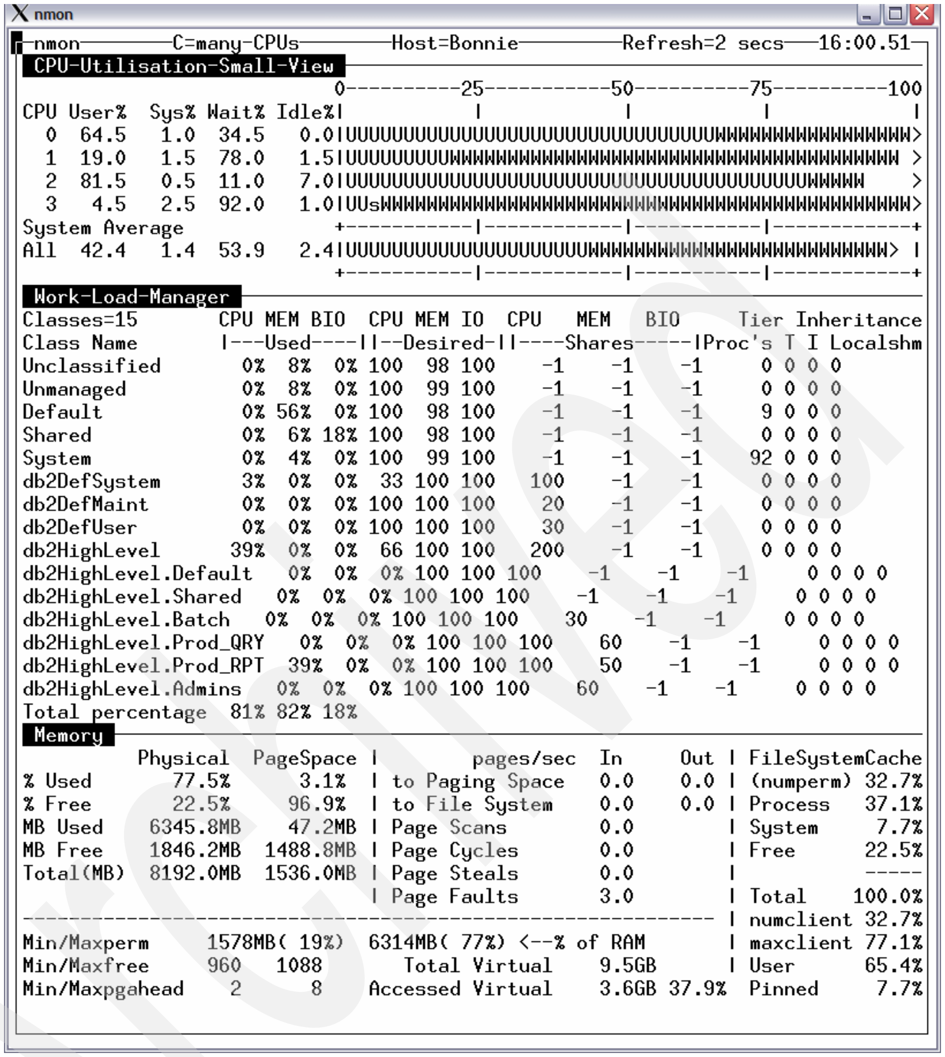
The **nmon** tool is a free tool for analyzing AIX performance. It is not part of the standard AIX operating environment and is not officially supported by IBM. The nmon tool has a user-friendly environment that is well-suited for real-time monitoring. Using **nmon**, you can see your CPU, disk I/O, and memory usage in real time on 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 monitor 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 real time in one screen, which is very useful in finding bottlenecks and solving performance problems on the database system.

Figure 6-12 on page 169 presents simple real time monitoring for WLM superclasses 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 6345.8 MB physical memory and have 1846.2 MB 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 could examine this data further, because we would probably want to know what our CPUs are waiting for.

We can also look at the AIX WLM configuration from **nmon** real time data, and find CPU shares and tiers for different DB2 service classes. By having all this information at one central place, we are able to see how our overall system is performing at a glance.



```
 nmon──────────C=many-CPUs─────────Host=Bonnie──────────Refresh=2 secs───16:00.51─
 CPU-Utilisation-Small-View ─────────────────────────────────────────────────────
                           0----------25-----------50----------75----------100
CPU User%  Sys% Wait% Idle%|           |            |           |            |
  0  64.5   1.0  34.5   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUWWWWWWWWWWWWWWWWW>
  1  19.0   1.5  78.0   1.5|UUUUUUUUUWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW >
  2  81.5   0.5  11.0   7.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUWWWWW    >
  3   4.5   2.5  92.0   1.0|UUsWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW>
System Average             +-----------|------------|-----------|------------+
All  42.4   1.4  53.9   2.4|UUUUUUUUUUUUUUUUUUUUUWWWWWWWWWWWWWWWWWWWWWWWWWWW> |
                           +-----------|------------|-----------|------------+
 Work-Load-Manager ──────────────────────────────────────────────────────────────
Classes=15        CPU MEM BIO  CPU MEM IO  CPU    MEM   BIO    Tier Inheritance
Class Name       |---Used----||--Desired-||----Shares-----|Proc's T I Localshm
Unclassified        0%  8%  0% 100  98 100    -1     -1    -1     0 0 0 0
Unmanaged           0%  8%  0% 100  99 100    -1     -1    -1     0 0 0 0
Default             0% 56%  0% 100  98 100    -1     -1    -1     9 0 0 0
Shared              0%  6% 18% 100  98 100    -1     -1    -1     0 0 0 0
System              0%  4%  0% 100  99 100    -1     -1    -1    92 0 0 0
db2DefSystem        3%  0%  0%  33 100 100   100     -1    -1     0 0 0 0
db2DefMaint         0%  0%  0% 100 100 100    20     -1    -1     0 0 0 0
db2DefUser          0%  0%  0% 100 100 100    30     -1    -1     0 0 0 0
db2HighLevel       39%  0%  0%  66 100 100   200     -1    -1     0 0 0 0
db2HighLevel.Default    0%  0%  0% 100 100 100    -1     -1    -1     0 0 0 0
db2HighLevel.Shared   0%  0%  0% 100 100 100    -1     -1    -1     0 0 0 0
db2HighLevel.Batch  0%  0%  0% 100 100 100    30     -1    -1     0 0 0 0
db2HighLevel.Prod_QRY   0%  0%  0% 100 100 100    60     -1    -1     0 0 0 0
db2HighLevel.Prod_RPT  39%  0%  0% 100 100 100    50     -1    -1     0 0 0 0
db2HighLevel.Admins   0%  0%  0% 100 100 100    60     -1    -1     0 0 0 0
Total percentage  81% 82% 18%
 Memory ─────────────────────────────────────────────────────────────────────────
          Physical  PageSpace |        pages/sec  In     Out  | FileSystemCache
% Used       77.5%      3.1%  | to Paging Space   0.0    0.0  | (numperm) 32.7%
% Free       22.5%     96.9%  | to File System    0.0    0.0  | Process   37.1%
MB Used   6345.8MB    47.2MB  | Page Scans        0.0         | System     7.7%
MB Free   1846.2MB  1488.8MB  | Page Cycles       0.0         | Free      22.5%
Total(MB) 8192.0MB  1536.0MB  | Page Steals       0.0         |           -----
                              | Page Faults       3.0         | Total    100.0%
------------------------------------------------------------  | numclient 32.7%
Min/Maxperm     1578MB( 19%)  6314MB( 77%) <--% of RAM        | maxclient 77.1%
Min/Maxfree     960   1088       Total Virtual    9.5GB       | User      65.4%
Min/Maxpgahead    2      8    Accessed Virtual    3.6GB 37.9% | Pinned     7.7%
```

Figure 6-12   Real-time monitoring service classes with nmon

Note that **nmon** is also capable of saving monitored data to file in a comma-separated values (CSV) format. It is an ideal tool for collecting data for analysis using charts. Many of our AIX WLM-related charts in this chapter were created using **nmon**.

In Figure 6-10 on page 164, we present a chart showing CPU usage by service classes. We noticed that there was a gap in 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 is then imported to a spreadsheet for plotting charts. When data is collected using operating system tools, then both DB2-related data and the entire operating system-related data is collected.

Monitoring system using AIX WLM and DB2 WLM monitoring

In Figure 6-10 on page 164, we presented a CPU usage by service classes chart using data from **nmon**. We noticed that there was a gap in CPU usage between 6:32 pm and 6:33 pm, and wanted to know the cause of it.

Because the data was collected with an operating system tool, the chart reflects the activities on the entire system, both 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 an 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 6-13.

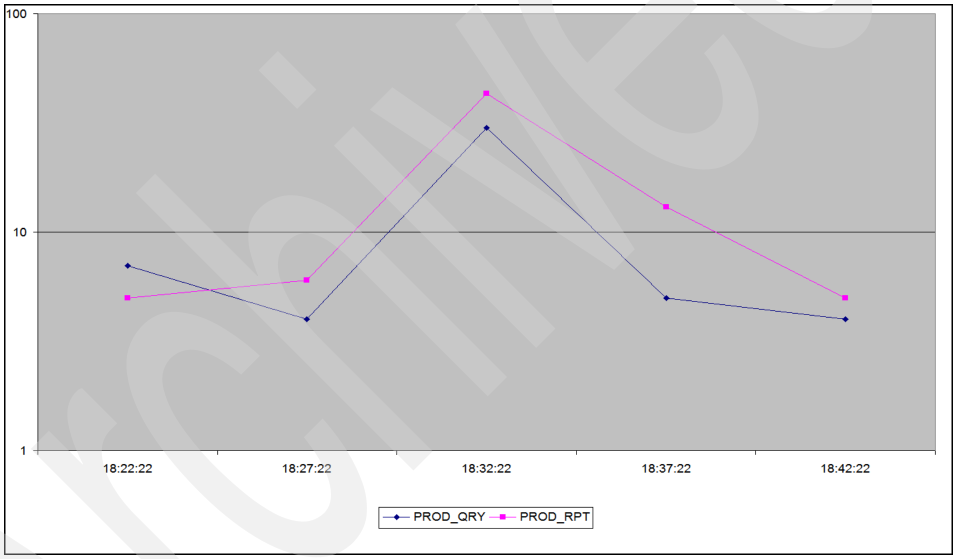


Figure 6-13 Average execution time

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 a lack of operating system resources.

We then examine the operating system disk I/O activities at the time of this incident by using the same data file collected by **nmon**. Figure 6-14 shows the disk reads between 6:26 pm and 6:40 pm.

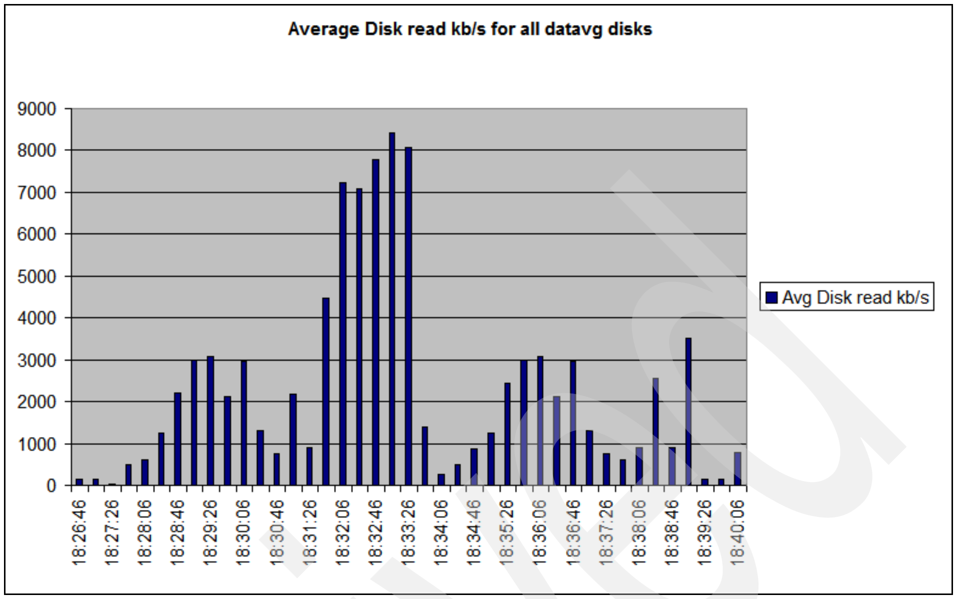


Figure 6-14 Operating system disk reads

The disk reads in the chart are average for all the disks that belong to the datavg volume group. This volume group holds all of our database-related data. There was a high disk read period between 6:32 pm and 6:33 pm because the operating system administrator mistakenly started the backup procedure for the volume group, then corrected this action quickly. DB2 queries were in I/O wait and did not consume CPU time.

We were able to determine the root cause for the relatively short service breakdown 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 optimal database performance.

7

WLM sample scenarios - other usage

This chapter demonstrates how to use WLM beyond simply managing DB2 workloads. Using the same data captured in the previous scenarios, WLM shows its usefulness in solving other tasks in running a data center.

In this chapter, we explore two sample scenarios:

- Capacity planning - how to use WLM to trend and project resource consumption and anticipate future resource needs, such as more CPU power
- Chargeback accounting- how to use WLM to establish a method of capturing resource utilization for the purposes of charging for resource usage

7.1 Capacity planning

Capacity planning has two main objectives:

- Trending - Establish resource (CPU, memory, DASD) usage and workload trends over an extended period of time
- Projecting - After 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 it can involve trending and projecting many resources. Our example here is not designed to replace existing capacity planning tools, but instead to augment them by providing additional data or insights about how resources are being consumed by DB2 applications.

For simplicity, our example covers a single resource: CPU. However, the principles can be applied to other resources, such as memory.

In this case, we want to know how much CPU resource we are using today so that we can project whether we have adequate resources tomorrow, when workloads change. By establishing trends on our DB2 applications, we can determine whether resource needs are changing over time.

Such projections and trends are needed when business grows and more data is added to our database; when new applications are added and the number of workloads will increase; or when more users are expected to use our 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 following DB2 WLM monitoring data is needed for specific time periods:

- 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 -rwlm
```

This creates an output file that can be input to nmon analyzer.xls. Both nmon and the nmon analyzer.xls are available for download from the following IBM Wiki Web site:

`http://www.ibm.com/collaboration/wiki/display/WikiPtype/nmon`

7.1.1 The workload environment

We start our capacity planning scenario by using the environment established in Chapter 3, "Customizing the WLM execution environments" on page 49. 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 7-1 on page 175. The other workloads are of no immediate concern and are ignored.

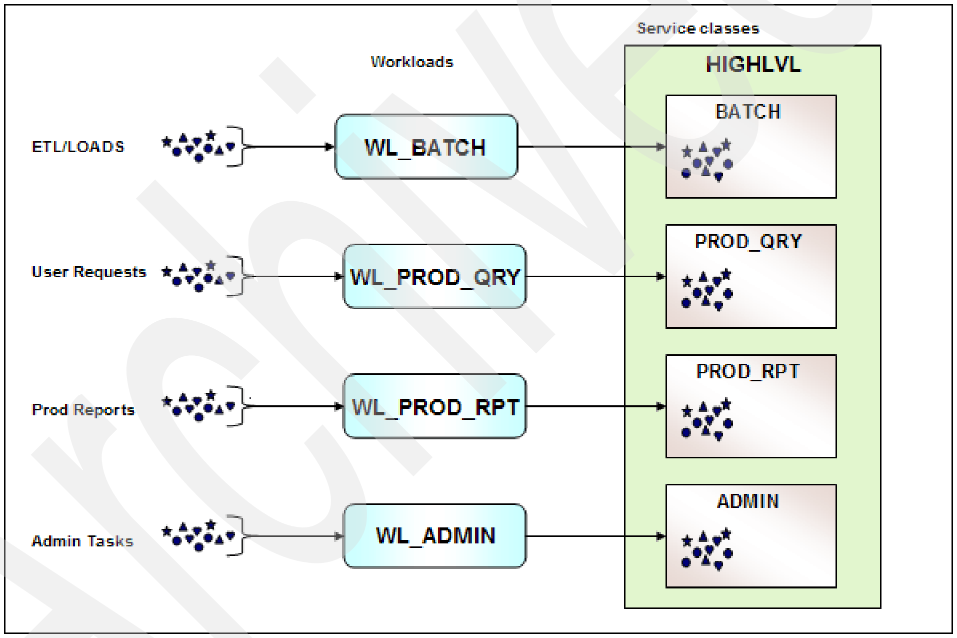


Figure 7-1 Workload environment (from Chapter 3)

The following monitoring requirements are needed for capacity planning:

- Retention period: 12 months
- Intervals for monitoring: 30 minutes

7.1.2 Collecting the trending data

During running sample workloads, the statistics information is captured and sent to the statistics event monitor at 30-minute intervals automatically. We use the SCSTATS_BASIC_MON table to monitor the base information for capacity planning.

For each interval, we want to know the highest number of concurrent queries running, and the combined execution time. The concurrent activity top shows us the highest concurrency of activities (including nested activities) during a specific interval, giving us a sense of how many workloads are taking place in our system for any given time period. Combining that with the total execution time for the same period, along with the nmon CPU usage, we can determine how much CPU is being consumed by our workloads.

The nmon CPU data must be used to confirm that the request execution times are the result of the queries running and consuming CPU, as opposed to waiting for locks or data from disks. By looking at the time intervals, we can find when our highest workload periods are occurring.

Assuming a continuance of the same workloads within the same time periods as measured over time, we can then analyze and project our capacity into the future. Again, it must be noted that these are simple techniques for a complex problem—but they can be effective for obtaining a high level view for capacity planning analysis.

7.1.3 Monitoring and analysis

To collect the data for our example, we used the query shown in Example 7-1

Example 7-1 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 decimal(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);
```

After collecting the data, we can create various charts for analysis. For this example, we have collected three weeks of data, covering the prime time hours.

Our first chart, shown in Figure 7-2 on page 178, 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 prime time hours are shown, because this is our main area of focus.

From this chart, we can see which hours have the highest request execution times, and whether a trend appears to be established. Here we clearly see that 11 am shows the highest total request execution times. Additionally, we appear to have established an upward trend in request execution times. Notice that the request execution time drops off dramatically after 5 pm, and continues to remain low for the rest of the prime time shift.

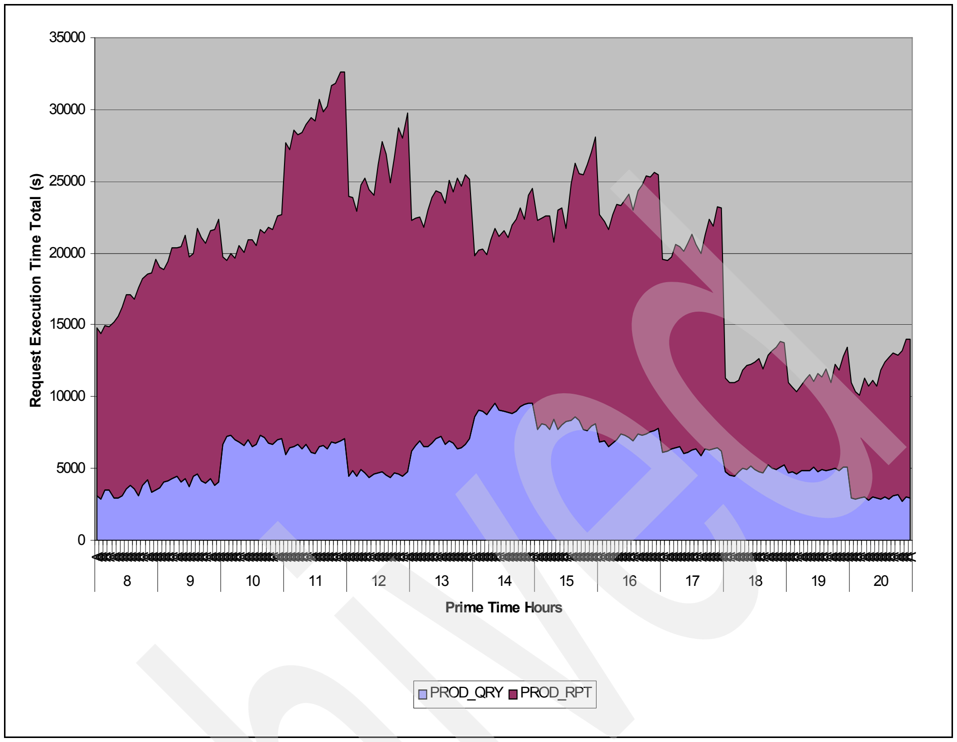


Figure 7-2 Total Request Execution times in seconds, across all partitions

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 on page 180, we see the total CPU percentage for all our partitions. This can easily be captured using **`vmstat`** or **`nmon`**.

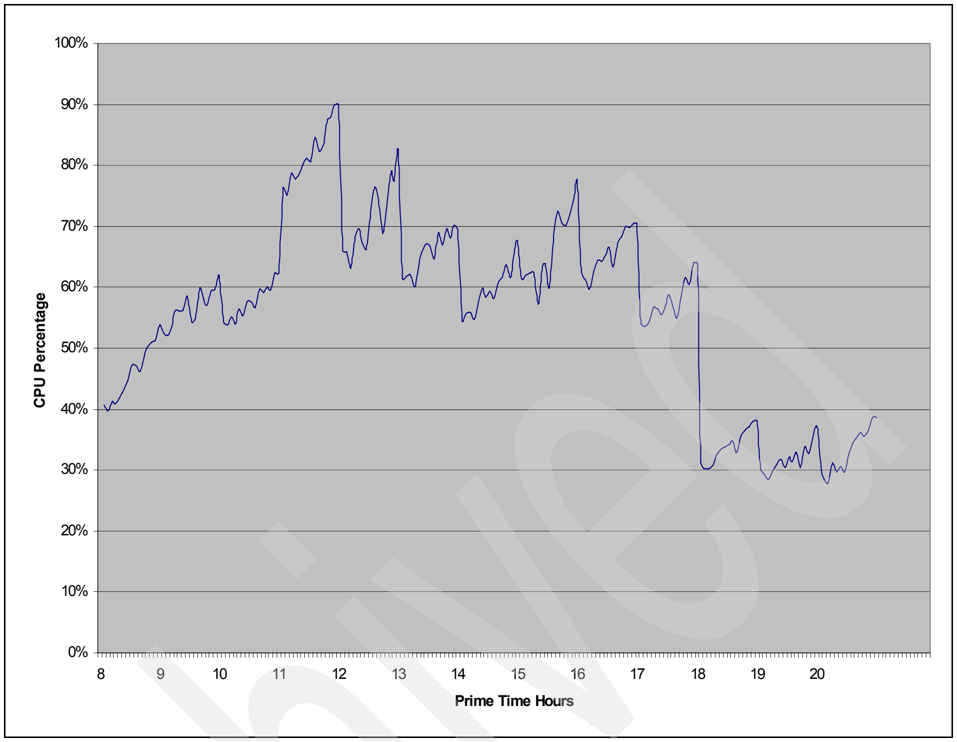


Figure 7-3 Total CPU percentages across all partitions

Now we can put the previous chart into perspective. 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. Therefore, we can make a basic assumption that we reach maximum consumption at 100% when we reach a total request execution time of 36,261 (32,635/.90) seconds. Other factors play into CPU consumption but for basic planning purposes, this gives a simple model.

Next, we need to analyze whether any correlation exists between request execution times and the number of active concurrent workloads. Figure 7-4 on page 180 displays a chart showing the number of top concurrent workload activities for the same time periods.

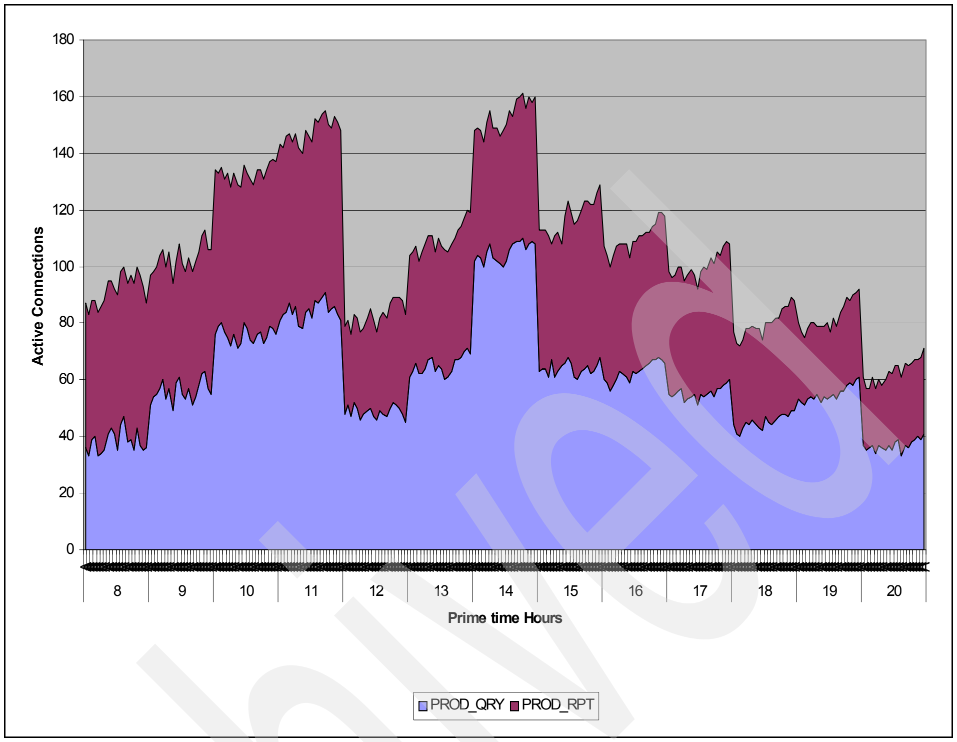


Figure 7-4 Number of top concurrent workloads activities for time periods

Here we see a slightly different picture: we have two periods of high workloads, 11 am and 2 pm. The difference must be in the average request execution times for each workload activity. Figure 7-5 on page 181 displays a chart showing the average request execution time for average concurrent workloads.

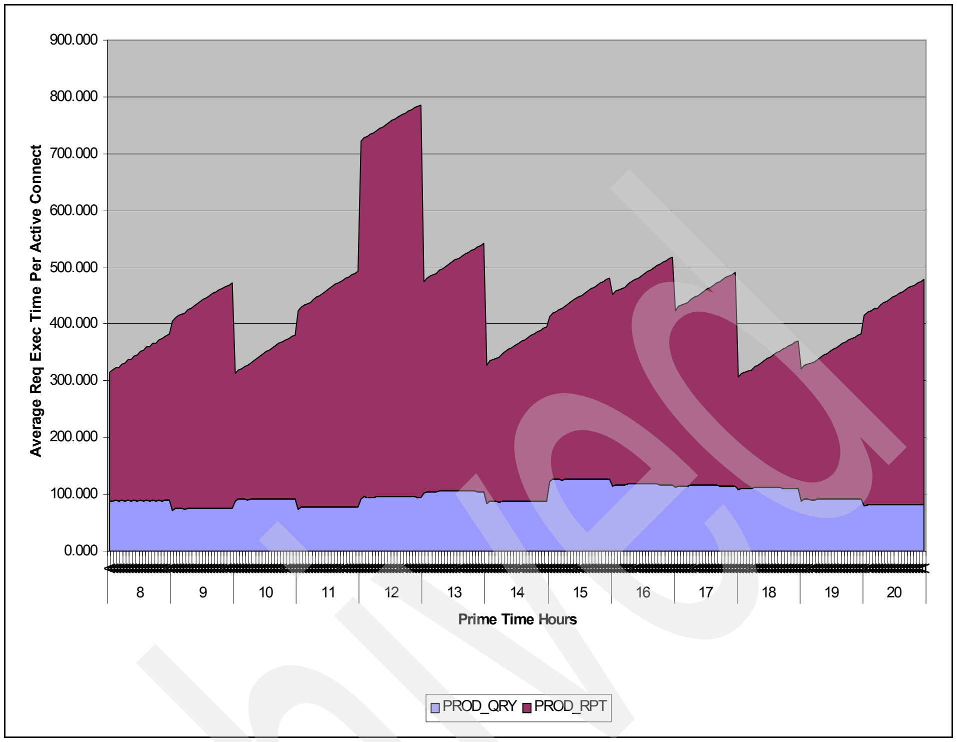


Figure 7-5 Average request execution times per active connection

As you can see, our most “expensive” requests are submitted during lunch time, at 12 pm. We can also confirm that the workloads at 2 pm are slightly more CPU-intensive then those submitted at 11 am.

All of our charts confirm that a linear correlation exists between Request Execution times and Percentage of CPU. As the total request execution time goes up, so does the total percentage of CPU.

This is important to establish and understand, because if there is no linear relationship between these two factors, it indicates that other factors may be at play. The other factors could be lock waits or slow disk access times, or perhaps there is not enough memory and paging is occurring. A more comprehensive set of reports with additional information would then be needed in order to effectively analyze these additional factors.

7.1.4 Capacity planning - summary

From our simple capacity planning example, you can see that WLM provides a effective means for performing capacity planning. Note that several assumptions were made in doing the analysis.

First, the linear correlation between the WLM request execution times and CPU utilization exists. When the WLM request executions times go up, so does the CPU utilization. If this relationship is not linear, then more complex mathematical analysis must be used to establish the proper relationship.

Secondly, 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 reveals several important facts:

- Our workload activity is not evenly distributed across our primtime shift.
- 11 am is our highest CPU demand period, due to the high number of requests.
- 12 pm is when our most CPU-intensive workloads are submitted. The workloads in this time period may need to be investigated to see whether 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 the need to expand our capacity.

7.2 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 section shows two instances of DB2, with each instance containing a database. This might be a typical OLTP environment where the two databases are in their own instances to allow for separate configurations. This arrangement also insures stability between databases, as well as the independence to start and stop DB2 when needed. We want each instance and database charged to its respective department.

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 explains how to use DB2 WLM for chargeback accounting. For information regarding AIX WLM, refer to Chapter 6, “AIX Workload Manager considerations” on page 143.

7.2.1 Business objectives

In our environment, we have two departments (Sales and Accounting) sharing the same server but running in separate instances. The two areas are responsible for paying for the entire system so the costs are split, based on their proportionate share. This includes times when the entire system may not be busy. This would be typical in a zero cost data center, where all the costs are allocated.

7.2.2 Defining workload profile

Example 7-2 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 7-2 WLM configuration for chargeback accounting

```
-- instance db2inst01
CREATE SERVICE CLASS HIGHLVL DISABLE;
CREATE SERVICE CLASS SALES UNDER HIGHLVL 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 HIGHLVL DISABLE;
CREATE SERVICE CLASS SALES UNDER HIGHLVL 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;
```

7.2.3 Monitoring

After we have run our WLM setup for a week, we can analyze our data. Using the query in Example 7-3, we can capture the chargeback data from each instance.

Because the event monitor names are the same but the super class names are unique, we simply run the query in both instances.

Example 7-3 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 Table 7-1 on page 184.

Table 7-1 Chargeback accounting spreadsheet

Total Request Execution Times				**Percentage chargeback**	
STATISTICS_DATE	**ACCOUNTING**	**SALES**	**Grand total**	**ACCOUNTING**	**SALES**
08/27/07	98,940.58	218,744.34	317,684.93	31.144	68.856

Total Request Execution Times				Percentage chargeback	
STATISTICS_DATE	**ACCOUNTING**	**SALES**	**Grand total**	**ACCOUNTING**	**SALES**
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

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, IT would be charged 100%. This eliminates having a shortfall or gap in accounting for the machine usage.

Because we are only using DB2 WLM Request Execution times, we have to assume that the other DB2 processes are not accounted for in WLM. Here 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

We can represent the spreadsheet in Table 7-1 on page 184 graphically to illustrate how the departments compare to each other; see Figure 7-6 on page 186.

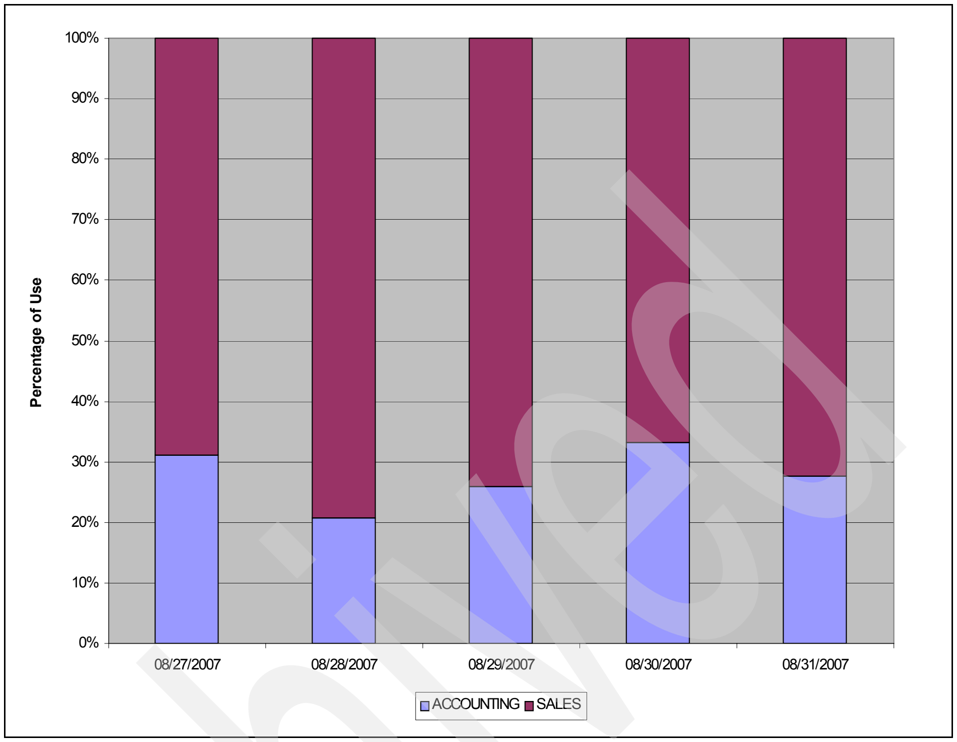


Figure 7-6 Chargeback accounting - graphic representation

7.2.4 Chargeback accounting - 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 do not adversely change the percentages. If it is determined that the processes which are unaccounted for are out of proportion, then the default super classes can be included in the chargeback accounting.

As mentioned, AIX WLM can be used on AIX systems for chargeback accounting, and the AIX WLM setup is shown in Chapter 6, "AIX Workload Manager considerations" on page 143.

8

DWE Design Studio and DB2 WLM

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.

We discuss the following topic in this chapter:

- A detailed description of how to create DB2 WLM components using DWE Design Studio

8.1 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 lets 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 used for building integrated development environments (IDEs). Eclipse is an open source community of companies that focuses on building development platforms, 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); create relationships between data elements (data mining); move and transform data within the data warehouse; analyze 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, such as the runtime environment, user interface, and help systems are 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 8-1.

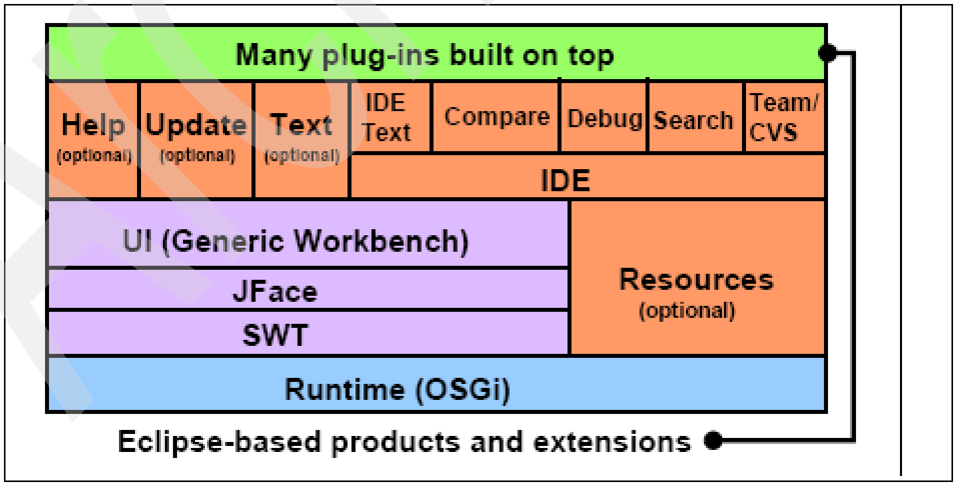


Figure 8-1 Eclipse basic platform

8.1.1 Workload management support

Prior to DB2 9.5, the workload management solution was based on Query Patroller (QP) and DB2 Governor. Query Patroller is a predictive tool that provides a way to monitor, manage, and control work, and provides reports. DB2 Governor is a reactive tool that uses a system monitor to watch the work running on the system and, based on the rules defined, takes reactive steps to correct problematic situations.

In DB2 9.5, a comprehensive workload management feature is integrated inside the DB2 engine to closely interact, access, and manage workloads so that you can see how your system is running, and 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, as described here:

- Reverse engineering on an existing database

  You can use Design Studio to reverse-engineer an existing DB2 WLM object's definitions, settings, and other information, in order to create a new WLM scheme. This scheme can be changed to suit your needs, and it can be validated. Reverse engineering allows you to: port the current database settings to Design Studio for further editing; port to another system; or use the database settings as the base for a new workload.

- Create new WLM solutions using templates

  Design Studio provides standard templates that can used to achieve your WLM objectives:

  - Provide resource sharing

    On a shared system environment, resources have to be shared. Using the WLM solution, you can control and share system resources based on the needs, priority, or other defined agreements. You can explicitly establish limits for the consumption of system resources; prevent the misuse of resources; and track resource usage patterns.

    For example, you can set up CPU sharing on AIX systems and, for other systems, set thread priorities of the agents and control the priority with which agents submit prefetch requests.

  - Enforce 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.

  - Enforce limits for concurrent activities

    You can enforce limits on the number of coordinator activities that run simultaneously. Limits can be enforced in the database, in the superclass, on the specific type of work, or on the specific type of work from a specific source.

- Manipulate WLM entities

  With Design Studio, the superclass, workloads, work actions, and thresholds can be viewed and changed by using the graphical interface.

- Validate 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 message output to alert you to the problems in the scheme that need to be corrected.

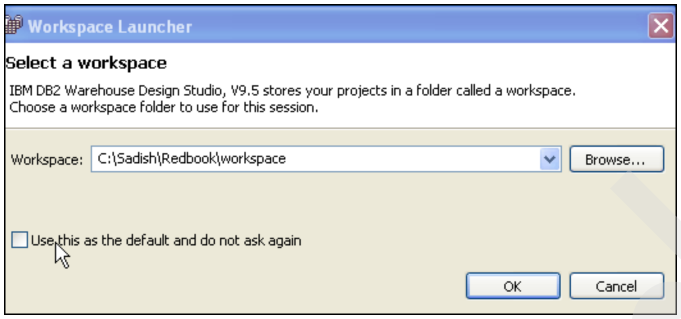
  On successful validation, you can generate the SQL code, review it, and revise the scheme to change the code. This can be repeated until you are satisfied that the scheme is ready to be deployed to the database.

- Deployment

  After a successful validation, you can deploy the workload management scheme directly from Design Studio by connecting to the database.

8.2 Getting started

After Design Studio is started, you are prompted to enter the workspace path as shown in Figure 8-2. 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.


Workspace Launcher

Select a workspace

IBM DB2 Warehouse Design Studio, V9.5 stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Sadish\Redbook\workspace

Figure 8-2 Workspace location

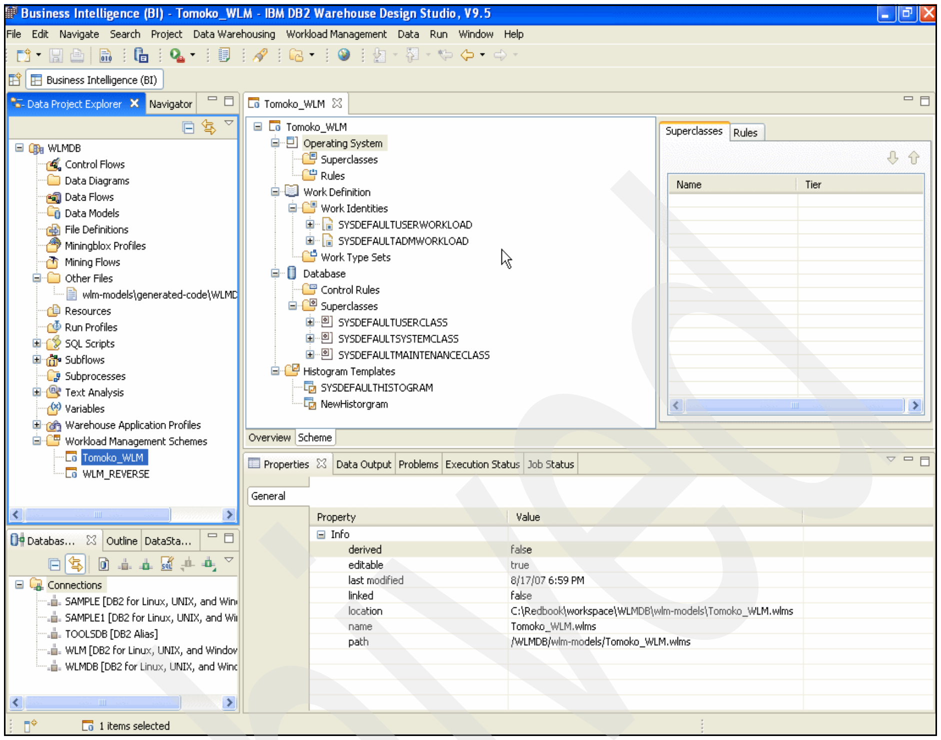
To switch between workspaces, select **File -> Switch Workspace**.

Note: We recommend that you select a workspace path which will be frequently backed up.

After you specify the workspace, Design Studio displays the Welcome view, which includes links to the DB2 Warehouse documentation, tutorials, and sample files.

Design Studio work is stored as projects, files, and folders. A *project* is a container used to organize resources pertaining to a specific subject area (for example, a data warehouse project). The workspace resources are displayed in a tree structure, with projects containing folders and files. Note that projects may *not* contain other projects.

Figure 8-3 shows the DWE Design Studio workbench.



Property	Value
Info	
derived	false
editable	true
last modified	8/17/07 6:59 PM
linked	false
location	C:\Redbook\workspace\WLMDB\wlm-models\Tomoko_WLM.wlms
name	Tomoko_WLM.wlms
path	/WLMDB/wlm-models/Tomoko_WLM.wlms

Figure 8-3 DWE Design Studio workbench

The perspectives control what the Design Studio displays, including certain menus and tool bars.

Perspectives

When you open the Design Studio, it takes you to the default Business Intelligence (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 your 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 the desired perspective.

To reset a perspective to its original layout, click **Window -> Reset Perspective**.

Views

A *view* is a visual component of Design Studio used 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, 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 Design Studio. This hierarchical tree displays projects and objects that you can navigate through or use to create new objects. This is a frequently used view, where you select and modify the contents of different projects and objects. Note that 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 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, picked up from your local catalog. You can set up connections to other databases as needed.

Properties view

This view is open by default in the lower right area of Design Studio. You can use the Properties view to define and modify many of the objects that you create. In addition, from this view you can see other views such as Data Output, Problems, Execution Status and Job Status. To make any of these views active, click the 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, verify that you have opened the correct view.

Editors view

The Editors view 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 has an object in a project. The available options are Text Editor, System Editor (operating system), and In-line Editor, based on the project scheme or objects.

Note: There is no automatic save option for these editors, so you must explicitly save the changes.

Projects

A *project* is a set of objects that you create in Design Studio as part of the data transformation or warehouse building processes. You must create a project in Design Studio before 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 the concurrent versions system (CVS). In a coordinated development environment, this will enable you to share the project with other developers.

8.2.1 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 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 terminology differences between the Workload Management Scheme on Design Studio and DB2 Workload Management. Table 8-1 lists the mapping of terminologies.

Table 8-1 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 using:

- The tree view
- The grid view

Figure 8-4 on page 195 shows a Design Studio tree view with WLM entities.

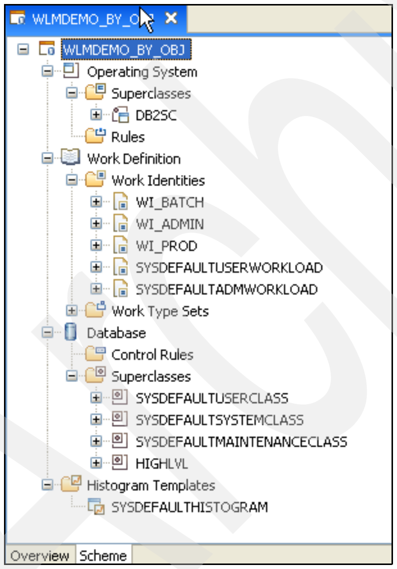
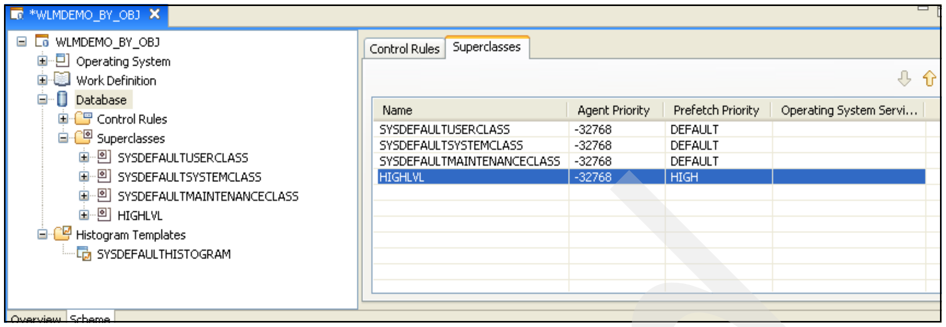


Figure 8-4 Design Studio WLM entities - tree view

Figure 8-5 shows the Design Studio WLM entities in grid view.



*WLMDEMO_BY_OBJ

- WLMDEMO_BY_OBJ
  - Operating System
  - Work Definition
  - Database
    - Control Rules
    - Superclasses
      - SYSDEFAULTUSERCLASS
      - SYSDEFAULTSYSTEMCLASS
      - SYSDEFAULTMAINTENANCECLASS
      - HIGHLVL
  - Histogram Templates
    - SYSDEFAULTHISTOGRAM

Control Rules | Superclasses

Name	Agent Priority	Prefetch Priority	Operating System Servi...
SYSDEFAULTUSERCLASS	-32768	DEFAULT	
SYSDEFAULTSYSTEMCLASS	-32768	DEFAULT	
SYSDEFAULTMAINTENANCECLASS	-32768	DEFAULT	
HIGHLVL	-32768	HIGH	

Figure 8-5 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 is used.

- 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; they must 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.

8.3 Managing database workloads using Design Studio

You can use 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 until 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 8-6.

In this example, we create a new project WLMDB.

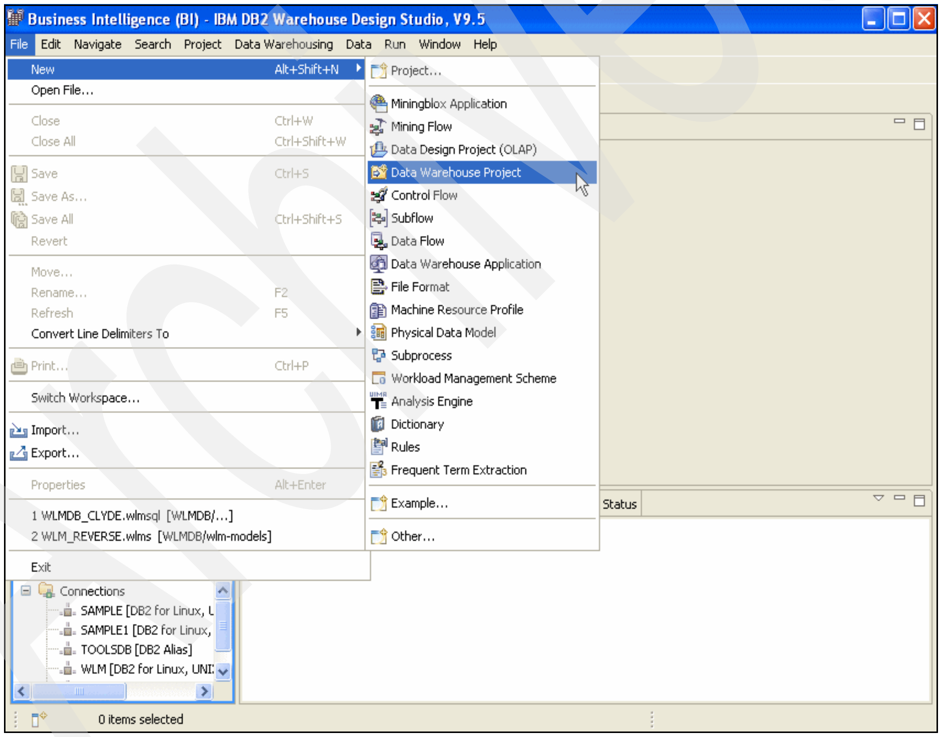
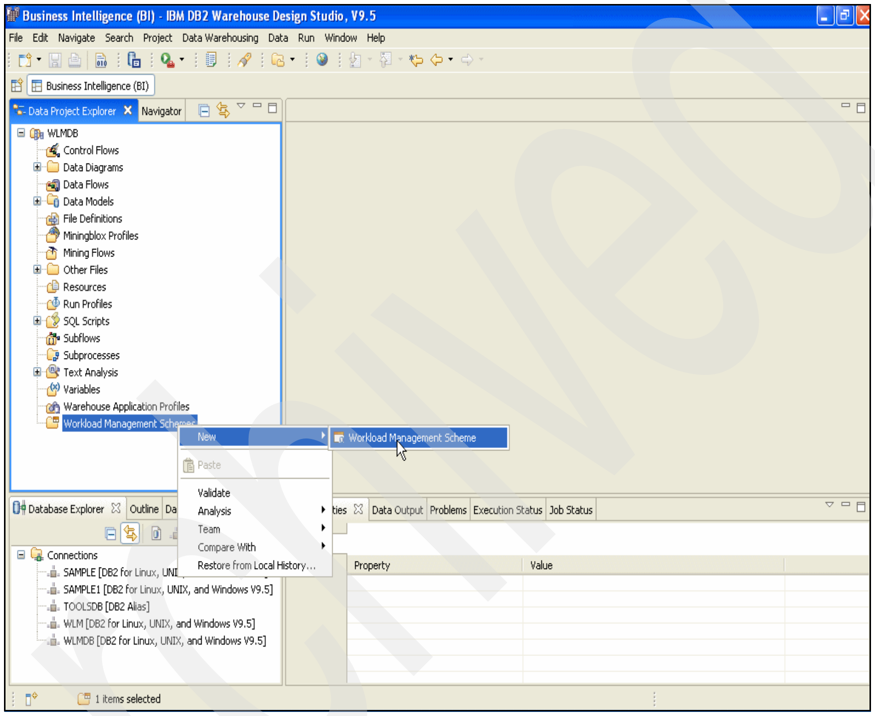


Figure 8-6 Create a new Data Warehouse project

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 creates a file with a .wlms file extension for the scheme, using the scheme name you specified under the workspace directory. Design Studio displays the complete path in the Overview tab.

To create WLM Scheme, expand the project tree view (WLMDB, in our example), and then right-click **Workload management scheme -> New -> Workload Management Scheme**; see Figure 8-7 on page 198**.**



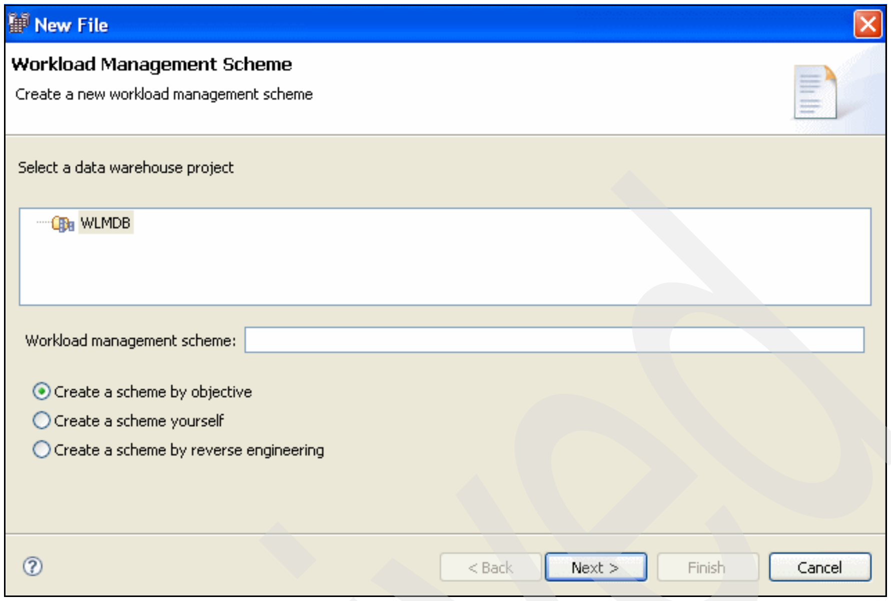
Business Intelligence (BI) - IBM DB2 Warehouse Design Studio, V9.5
File Edit Navigate Search Project Data Warehousing Data Run Window Help
Business Intelligence (BI)
Data Project Explorer
Navigator
WLMDB
Control Flows
Data Diagrams
Data Flows
Data Models
File Definitions
Miningblox Profiles
Mining Flows
Other Files
Resources
Run Profiles
SQL Scripts
Subflows
Subprocesses
Text Analysis
Variables
Warehouse Application Profiles
Workload Management Schemes
New
Workload Management Scheme
Paste
Validate
Analysis
Team
Compare With
Restore from Local History...
Database Explorer
Outline
Connections
SAMPLE [DB2 for Linux, UNIX, and Windows V9.5]
SAMPLE1 [DB2 for Linux, UNIX, and Windows V9.5]
TOOLSDB [DB2 Alias]
WLM [DB2 for Linux, UNIX, and Windows V9.5]
WLMDB [DB2 for Linux, UNIX, and Windows V9.5]
Data Output
Problems
Execution Status
Job Status
Property
Value
1 items selected

Figure 8-7 Creating workload management scheme

When you select New Workload Management Scheme, it takes you to the New File for Workload Management Scheme panel (Figure 8-8 on page 199), 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 creation method in detail in the following sections.


New File

Workload Management Scheme

Create a new workload management scheme

Select a data warehouse project

WLMDB

Workload management scheme:

(●) Create a scheme by objective
( ) Create a scheme yourself
( ) Create a scheme by reverse engineering

Figure 8-8 Selecting Create a new workload management scheme method

8.3.1 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 allow them to continue.

Controlling and sharing system resources

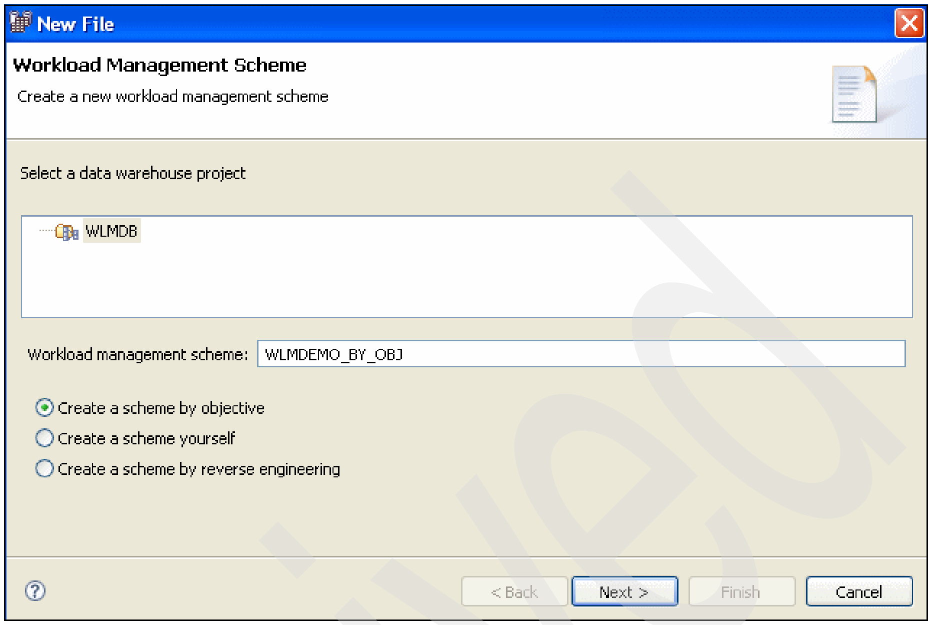
In this section, we demonstrate how to use the Control and Share system resources method to create a workload scheme. The example business problem used here is a database system which has short-, medium-, and long-running queries coming from different business units and competing for resources. We need to define a method so that work is grouped by business unit and shares system resources among each other in a controlled way.

One solution for this business case is the following:

- Categorize work and create appropriate work identities.
- Based on business units, create superclasses.
- Define relationships and assign different types of work identities to the 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 systems, use Agent Priority.

This proposed solution can be achieved by using the 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 the New File Workload Management Scheme panel; see Figure 8-9 on page 201.



Workload Management Scheme

Create a new workload management scheme

Select a data warehouse project

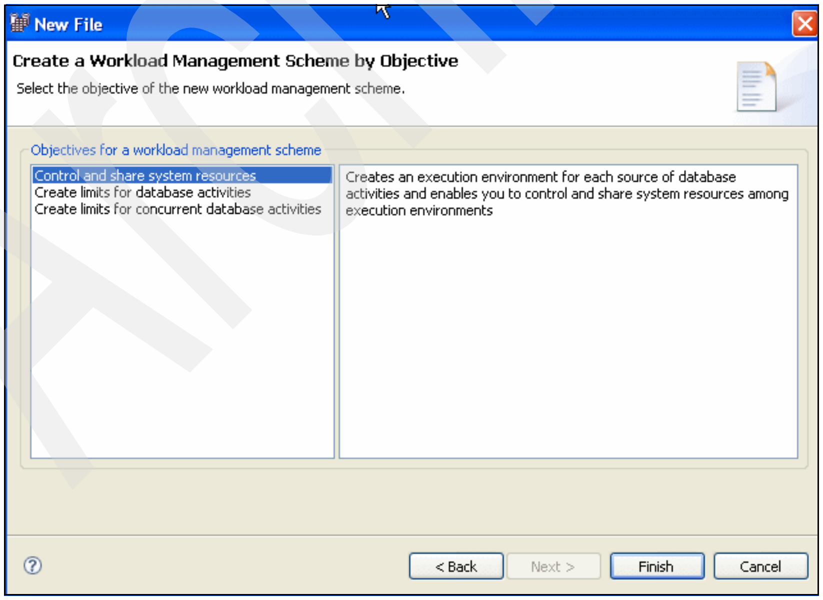
WLMDB

Workload management scheme: WLMDEMO_BY_OBJ

[✓ Create a scheme by objective]
Create a scheme yourself
Create a scheme by reverse engineering

Figure 8-9 Create scheme by objective

In the Create a Workload Management Scheme by Objective panel (Figure 8-10 on page 201), select **Control and share system resources** and click **Finish**.



Create a Workload Management Scheme by Objective

Select the objective of the new workload management scheme.

Objectives for a workload management scheme

[✓ Control and share system resources]
Create limits for database activities
Create limits for concurrent database activities

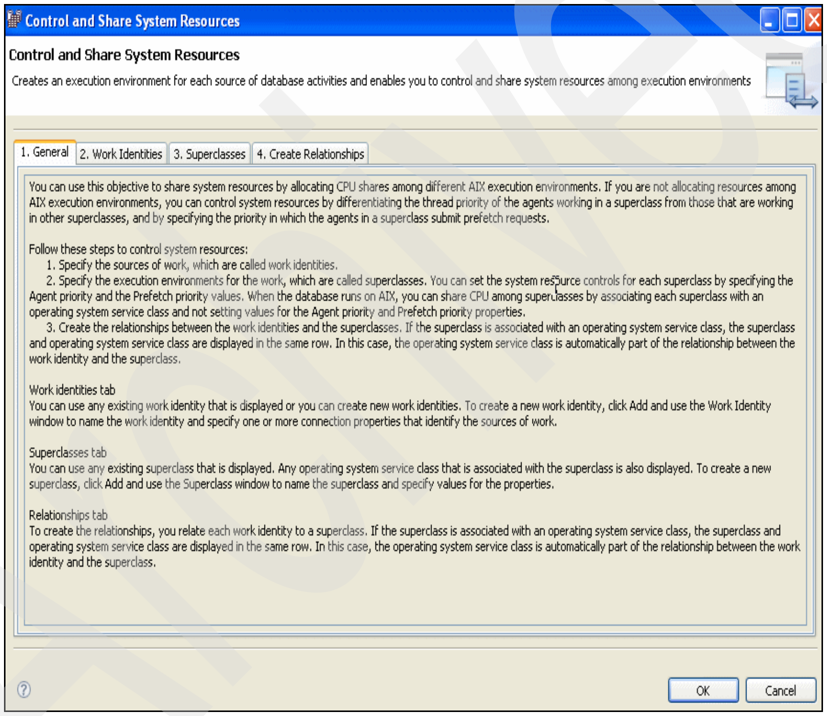
Creates an execution environment for each source of database activities and enables you to control and share system resources among execution environments

Figure 8-10 Control and share system resources

The Control and Share System Resources panel (Figure 8-11 on page 202) is then presented with four tabs labeled General, Work Identities, Superclasses, and Create Relationships. Each tab allows you to create specific DB2 workload objects.

Note: Clicking **OK** at any tab takes you back to the Business Intelligence view, and not to the next tab. Design Studio will create only the default objects for those unvisited tabs.

So to continue working on the scheme using the guided configuration, select the WLM scheme **-> Workload Management**, and then select the guided configuration you want.



Control and Share System Resources

Creates an execution environment for each source of database activities and enables you to control and share system resources among execution environments

You can use this objective to share system resources by allocating CPU shares among different AIX execution environments. If you are not allocating resources among AIX execution environments, you can control system resources by differentiating the thread priority of the agents working in a superclass from those that are working in other superclasses, and by specifying the priority in which the agents in a superclass submit prefetch requests.

Follow these steps to control system resources:
1. Specify the sources of work, which are called work identities.
2. Specify the execution environments for the work, which are called superclasses. You can set the system resource controls for each superclass by specifying the Agent priority and the Prefetch priority values. When the database runs on AIX, you can share CPU among superclasses by associating each superclass with an operating system service class and not setting values for the Agent priority and Prefetch priority properties.
3. Create the relationships between the work identities and the superclasses. If the superclass is associated with an operating system service class, the superclass and operating system service class are displayed in the same row. In this case, the operating system service class is automatically part of the relationship between the work identity and the superclass.

Work identities tab
You can use any existing work identity that is displayed or you can create new work identities. To create a new work identity, click Add and use the Work Identity window to name the work identity and specify one or more connection properties that identify the sources of work.

Superclasses tab
You can use any existing superclass that is displayed. Any operating system service class that is associated with the superclass is also displayed. To create a new superclass, click Add and use the Superclass window to name the superclass and specify values for the properties.

Relationships tab
To create the relationships, you relate each work identity to a superclass. If the superclass is associated with an operating system service class, the superclass and operating system service class are displayed in the same row. In this case, the operating system service class is automatically part of the relationship between the work identity and the superclass.

Figure 8-11 Control and Share resources - tab view

We explain the tabs as follows:

1. General

   This is an informational tab. Select the Work Identities tab to continue.

2. Work Identities

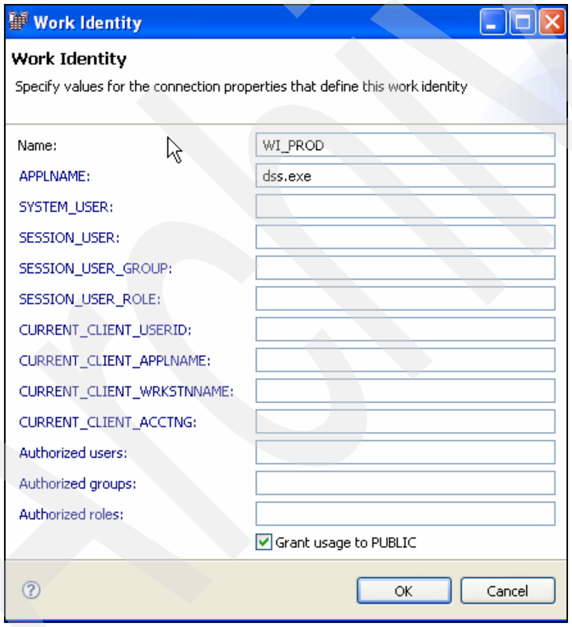
   You can use this tab to add a new DB2 WLM workload, or to delete or modify an existing workload.

   Because 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 was presented; see Figure 8-12 on page 203. You can identify the connection attributes to be used in your workload.

   To see the description of a field, left-click 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 workload execution authority.

   In our case, we created a workload WI_PROD to manage application dss.exe. The workload can be used by everyone (public). We selected the **Superclasses** tab to proceed.



Work Identity

Specify values for the connection properties that define this work identity

Name:	WI_PROD
APPLNAME:	dss.exe
SYSTEM_USER:	
SESSION_USER:	
SESSION_USER_GROUP:	
SESSION_USER_ROLE:	
CURRENT_CLIENT_USERID:	
CURRENT_CLIENT_APPLNAME:	
CURRENT_CLIENT_WRKSTNNAME:	
CURRENT_CLIENT_ACCTNG:	
Authorized users:	
Authorized groups:	
Authorized roles:	

✓ Grant usage to PUBLIC

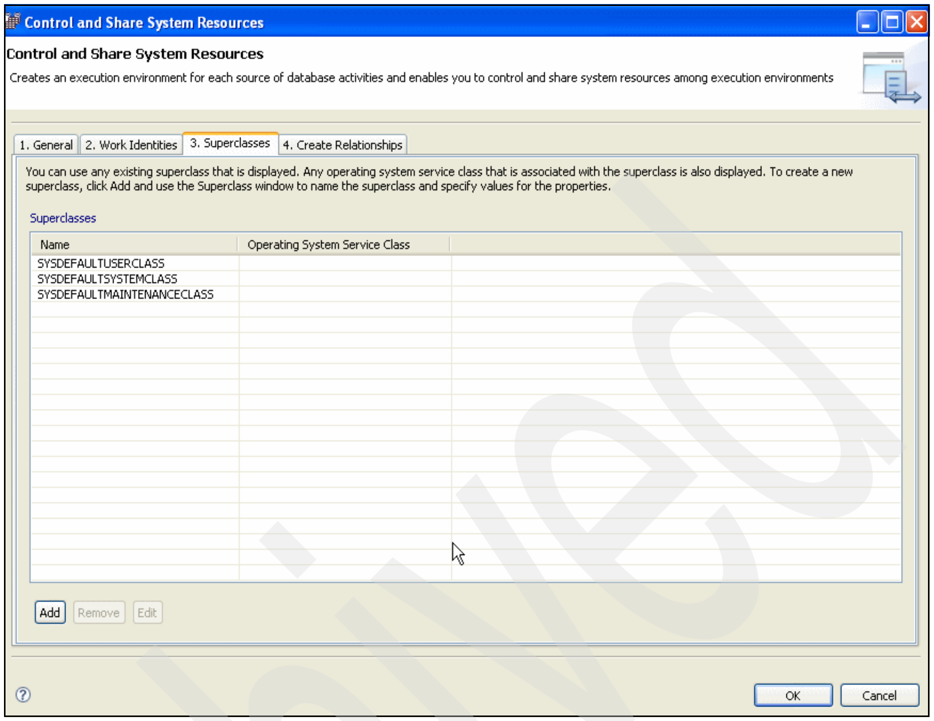
Figure 8-12 Work Identity

Notes:

- 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), then 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 the 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 8-13 on page 205 shows the Superclasses tab with three default superclasses: SYSDEFAULTUSERCLASS, SYSDEFAULTSYSTEMCLASS, and SYSDEFAULTMAINTENANCELASS. Use this tab to add new superclasses, or to delete or modify existing superclasses.


Control and Share System Resources

Control and Share System Resources

Creates an execution environment for each source of database activities and enables you to control and share system resources among execution environments

You can use any existing superclass that is displayed. Any operating system service class that is associated with the superclass is also displayed. To create a new superclass, click Add and use the Superclass window to name the superclass and specify values for the properties.

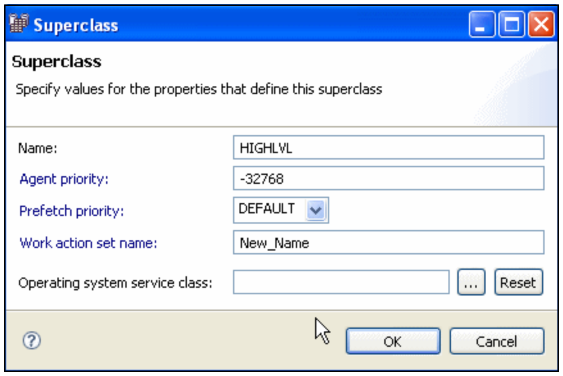
Superclasses

Name	Operating System Service Class
SYSDEFAULTUSERCLASS	
SYSDEFAULTSYSTEMCLASS	
SYSDEFAULTMAINTENANCECLASS	

Figure 8-13 Control and Share System Resources - Superclasses view

To create a new superclass, click **Add** and the Superclass property view will be presented; see Figure 8-14 on page 206.

Here you enter the name of the superclass to be created, the agent and prefetch priority, and work action set name, if already known. These fields match the AGENT PRIORITY, PREFETCH PRIORITY in the DB2 CREATE SERVICE CLASS statement.


Superclass

Superclass

Specify values for the properties that define this superclass

Name: HIGHLVL

Agent priority: -32768

Prefetch priority: DEFAULT

Work action set name: New_Name

Operating system service class:

Figure 8-14 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 8-15).

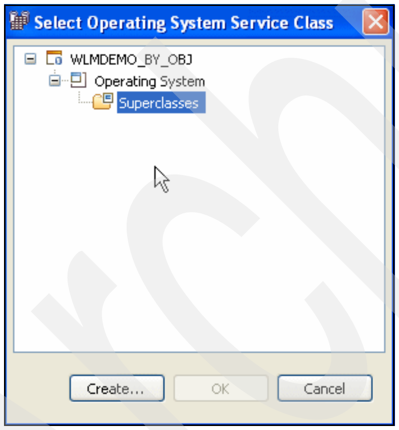
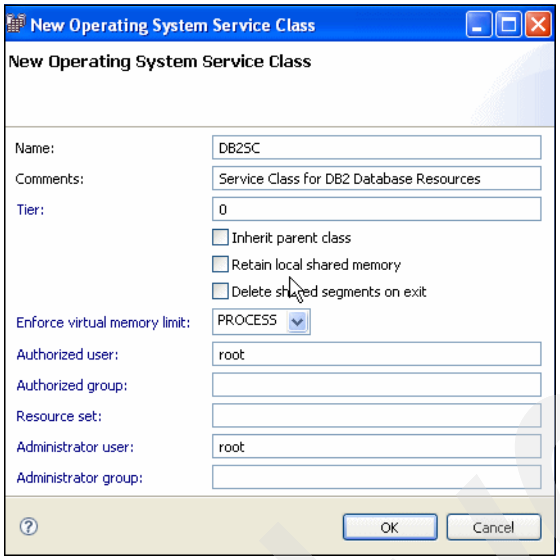


Figure 8-15 Select Operating system service class

To create a new operating system service class, expand the WLM scheme, select **Superclass**, and click **Create...**. For this example, we created an operating system class DB2SC for DB2 resources; see Figure 8-16. Clicking **OK** takes you back to the Select Operating System Service Class to allow you to create more operating system service classes.

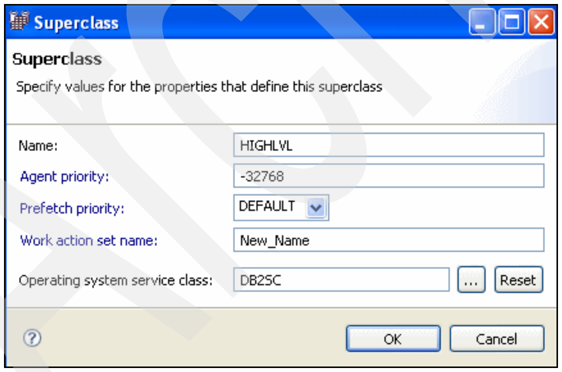

New Operating System Service Class

New Operating System Service Class

Name:	DB2SC
Comments:	Service Class for DB2 Database Resources
Tier:	0
	☐ Inherit parent class
	☐ Retain local shared memory
	☐ Delete shared segments on exit
Enforce virtual memory limit:	PROCESS
Authorized user:	root
Authorized group:	
Resource set:	
Administrator user:	root
Administrator group:	

Figure 8-16 New Operating System Service Class

After you have completed creating operating system service classes, select any operating system service class to make the OK button active and then click **OK**. Design Studio takes you back to Superclass property view showing the new operating system service class associated with DB2 Superclass; see Figure 8-17.


Superclass

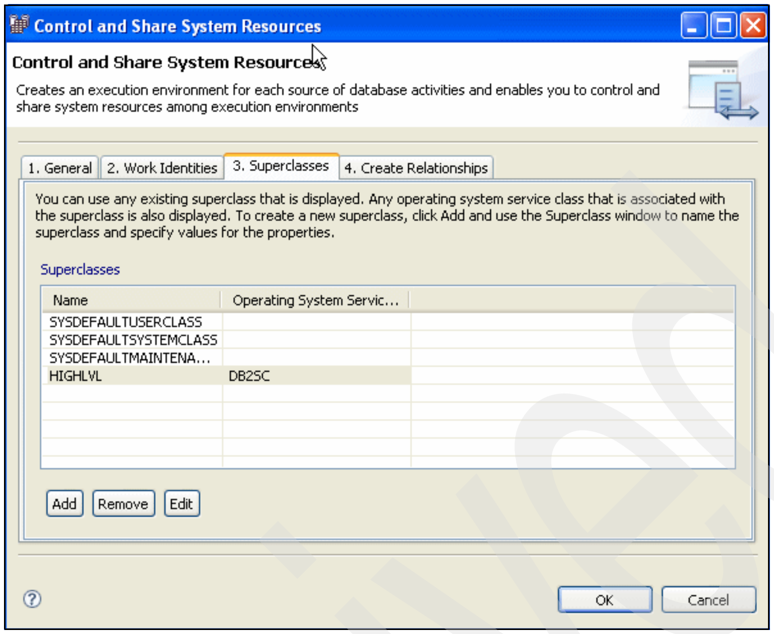
Superclass

Specify values for the properties that define this superclass

Name:	HIGHLVL
Agent priority:	-32768
Prefetch priority:	DEFAULT
Work action set name:	New_Name
Operating system service class:	DB2SC

Figure 8-17 DB2 Superclass with AIX Operating System Service class

Clicking **OK** in the Superclass property view takes you back to the Superclasses tab view for creating or editing another superclass. Figure 8-18 shows the DB2 superclass associated with AIX superclass. Clicking **OK** in this view ends the workload creating process. To continue, click the **Create Relationships** tab.



Control and Share System Resources

Control and Share System Resources

Creates an execution environment for each source of database activities and enables you to control and share system resources among execution environments

1. General | 2. Work Identities | 3. Superclasses | 4. Create Relationships

You can use any existing superclass that is displayed. Any operating system service class that is associated with the superclass is also displayed. To create a new superclass, click Add and use the Superclass window to name the superclass and specify values for the properties.

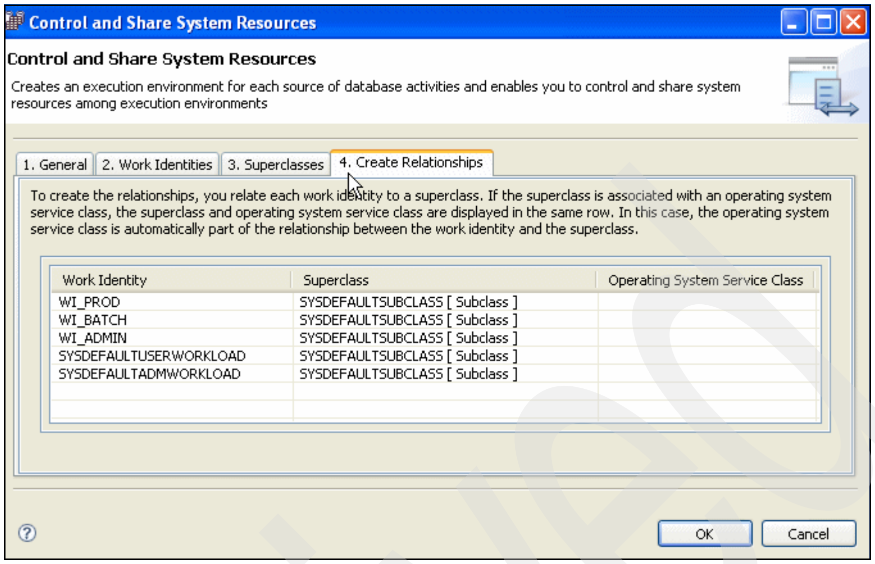
Superclasses

Name	Operating System Servic...
SYSDEFAULTUSERCLASS	
SYSDEFAULTSYSTEMCLASS	
SYSDEFAULTMAINTENA...	
HIGHLVL	DB2SC

Figure 8-18 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 8-19 on page 209 shows all the defined workloads are listed under Work Identity column.



Control and Share System Resources

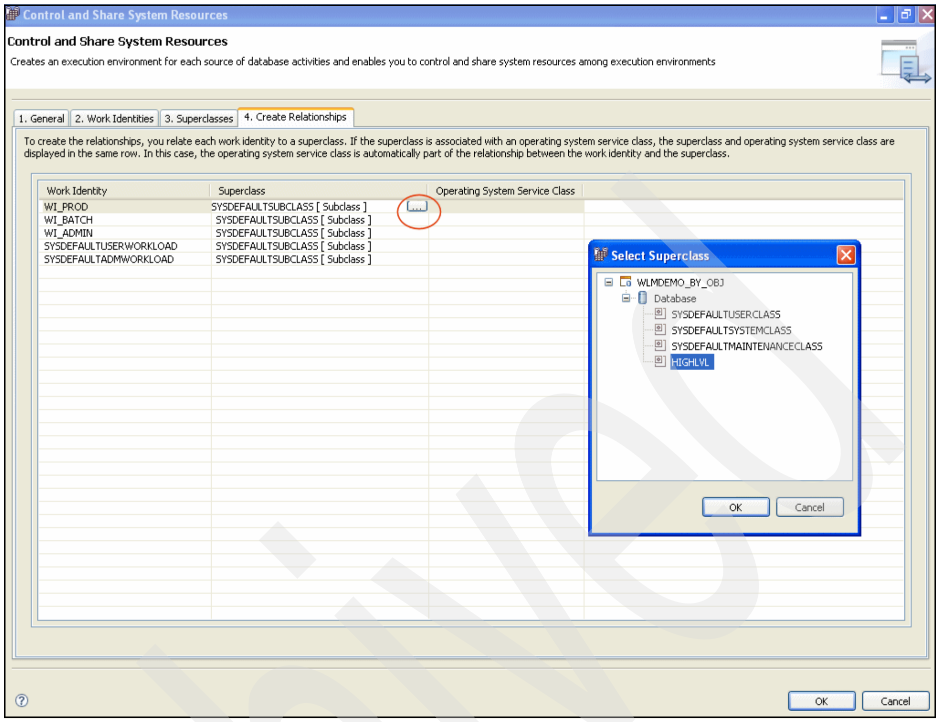
Creates an execution environment for each source of database activities and enables you to control and share system resources among execution environments

To create the relationships, you relate each work identity to a superclass. If the superclass is associated with an operating system service class, the superclass and operating system service class are displayed in the same row. In this case, the operating system service class is automatically part of the relationship between the work identity and the superclass.

Work Identity	Superclass	Operating System Service Class
WI_PROD	SYSDEFAULTSUBCLASS [Subclass]	
WI_BATCH	SYSDEFAULTSUBCLASS [Subclass]	
WI_ADMIN	SYSDEFAULTSUBCLASS [Subclass]	
SYSDEFAULTUSERWORKLOAD	SYSDEFAULTSUBCLASS [Subclass]	
SYSDEFAULTADMWORKLOAD	SYSDEFAULTSUBCLASS [Subclass]	

Figure 8-19 Create Relationships view

To change the superclass associated to a workload, click the superclass name; a browse button will be shown next to the superclass name. Click the browse button to obtain a list of superclasses you can choose from; see Figure 8-20 on page 210.



Control and Share System Resources

Creates an execution environment for each source of database activities and enables you to control and share system resources among execution environments

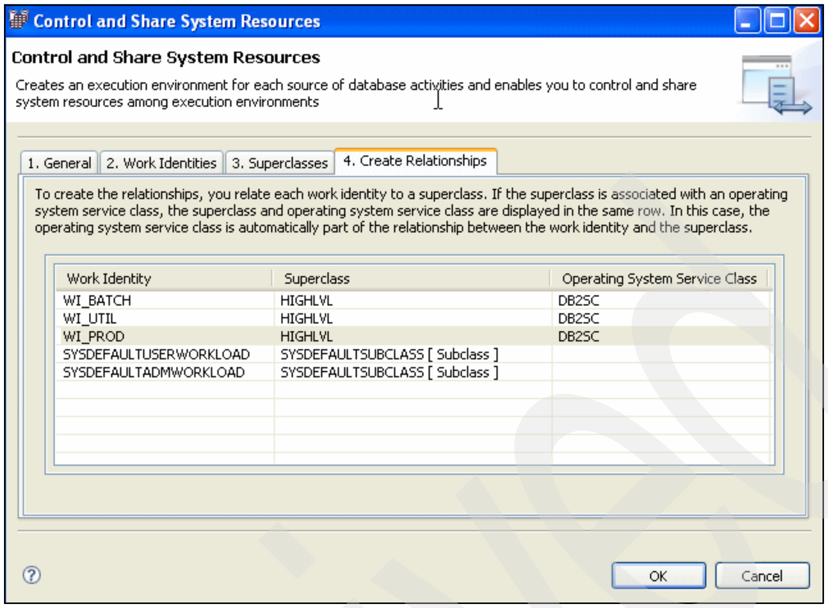
To create the relationships, you relate each work identity to a superclass. If the superclass is associated with an operating system service class, the superclass and operating system service class are displayed in the same row. In this case, the operating system service class is automatically part of the relationship between the work identity and the superclass.

Work Identity	Superclass	Operating System Service Class
WI_PROD	SYSDEFAULTSUBCLASS [Subclass]	
WI_BATCH	SYSDEFAULTSUBCLASS [Subclass]	
WI_ADMIN	SYSDEFAULTSUBCLASS [Subclass]	
SYSDEFAULTUSERWORKLOAD	SYSDEFAULTSUBCLASS [Subclass]	
SYSDEFAULTADMWORKLOAD	SYSDEFAULTSUBCLASS [Subclass]	

Figure 8-20 Create Relationships - associate with superclass

If you have a DB2 superclass that 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. After we associate the WI_PROD to HIGHLVL, the operating system service class DB2SC is automatically associated; see Figure 8-21 on page 211.



Control and Share System Resources

Control and Share System Resources

Creates an execution environment for each source of database activities and enables you to control and share system resources among execution environments

To create the relationships, you relate each work identity to a superclass. If the superclass is associated with an operating system service class, the superclass and operating system service class are displayed in the same row. In this case, the operating system service class is automatically part of the relationship between the work identity and the superclass.

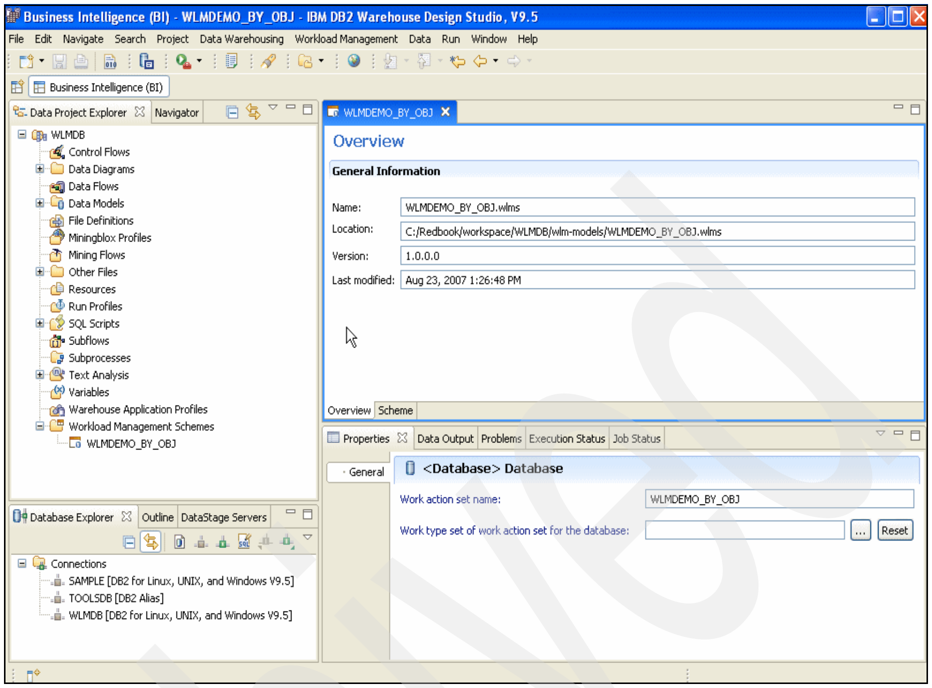
Work Identity	Superclass	Operating System Service Class
WI_BATCH	HIGHLVL	DB2SC
WI_UTIL	HIGHLVL	DB2SC
WI_PROD	HIGHLVL	DB2SC
SYSDEFAULTUSERWORKLOAD	SYSDEFAULTSUBCLASS [Subclass]	
SYSDEFAULTADMWORKLOAD	SYSDEFAULTSUBCLASS [Subclass]	

Figure 8-21 Control and Share System Resources - Create Relationships view

When you click **OK**, Design Studio creates a WLM scheme and takes 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. The project created for this example is WLMDEMO_BY_OBJ, as shown in Figure 8-22 on page 212.

Note: Design Studio does not save the scheme automatically. To save your scheme, use **File -> Save** or the icon.



Name: WLMDEMO_BY_OBJ.wlms
Location: C:/Redbook/workspace/WLMDB/wlm-models/WLMDEMO_BY_OBJ.wlms
Version: 1.0.0.0
Last modified: Aug 23, 2007 1:26:48 PM

Work action set name: WLMDEMO_BY_OBJ
Work type set of work action set for the database:

Figure 8-22 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; refer to Figure 8-23 on page 213. From the tree view, you can further modify the work scheme you created.

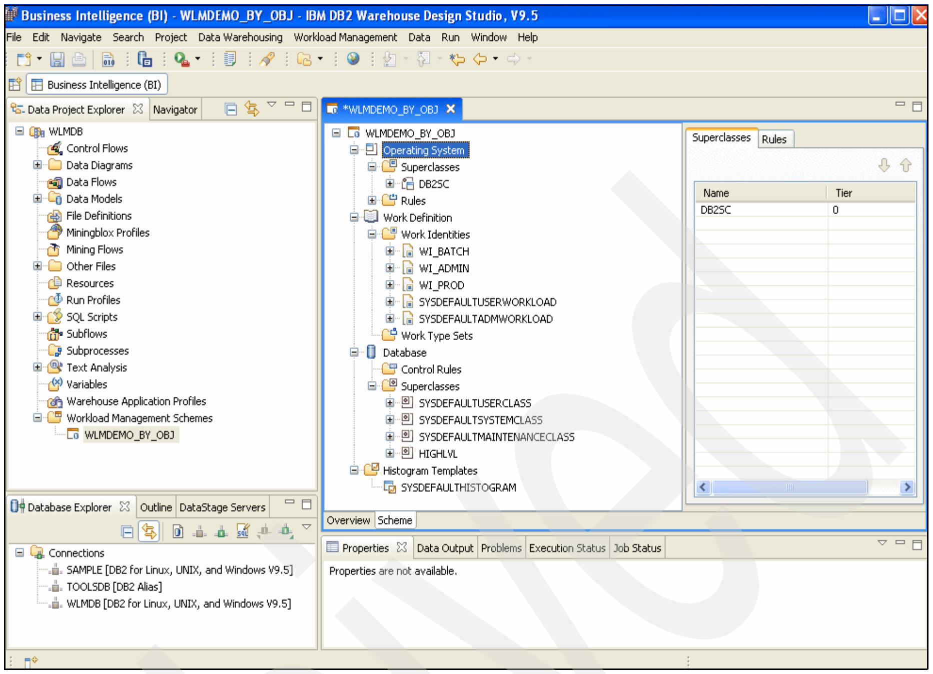


Figure 8-23 WLMDEMO_BY_OBJ Scheme view

Evaluation order of Work Identities

There is an evaluation order for DB2 workloads. When a database request arrives, 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.

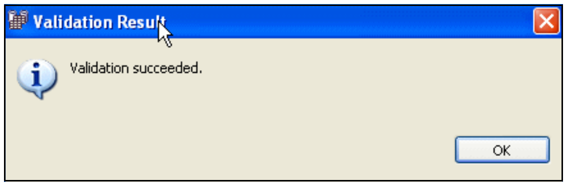
In DB2, the default workload position is *last*. When you create work identities using Design Studio, it generates the code with POSITION AT *position*, based on the order that the work identities are created. We recommend that you verify the workload sequence before creating the workloads.

To adjust the evaluation order, open the Work Identities tab by clicking **Work Definitions** in the Scheme view. Select the work identity in the grid view to the right of the tree view and use the up and down arrows to reposition it in the list.

Validating

After 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 validates 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 8-24 on page 214.



Validation succeeded.

Figure 8-24 Validation successful

Generating code

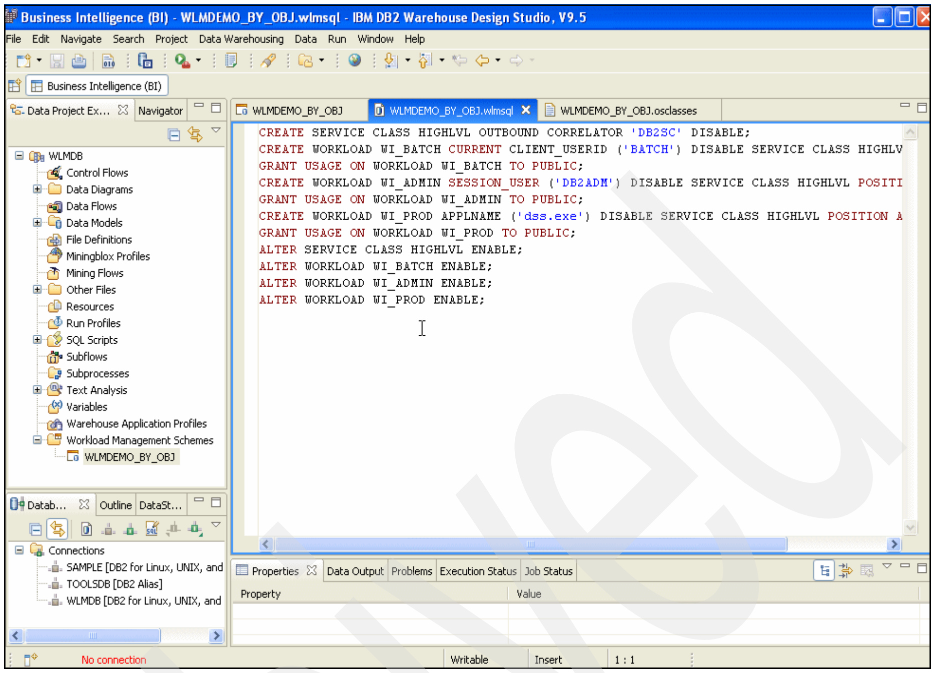
After successful validation, you can generate code using **Workload Management -> Generate Code**. For each code file generated, Design Studio presents it as a tab in the BI perspective Editors view.

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 8-25 on page 215 shows the generated WLMDEMO_BY_OBJ.wlmsql creating DB2 Workloads.

Although the Editors view allows you to add, modify, or remove the DB2 WLM statements, we do not recommend that you modify the code directly, because the modifications will *not* be captured in Design Studio. To capture the modifications in Design Studio, you need to perform reverse engineering after the workload is created in DB2.



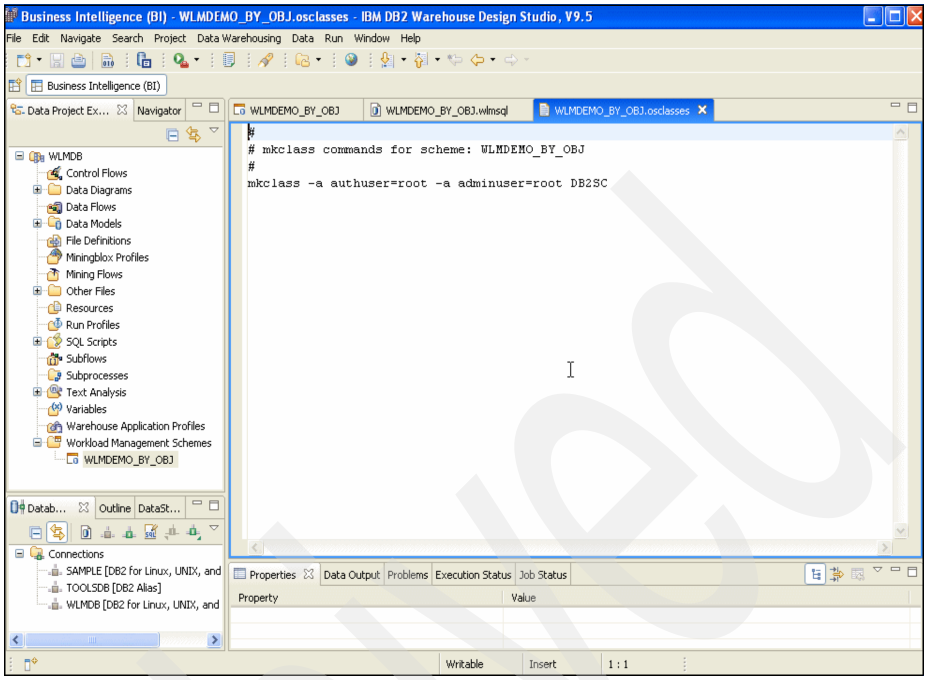
```
CREATE SERVICE CLASS HIGHLVL OUTBOUND CORRELATOR 'DB2SC' DISABLE;
CREATE WORKLOAD WI_BATCH CURRENT CLIENT_USERID ('BATCH') DISABLE SERVICE CLASS HIGHLV
GRANT USAGE ON WORKLOAD WI_BATCH TO PUBLIC;
CREATE WORKLOAD WI_ADMIN SESSION_USER ('DB2ADM') DISABLE SERVICE CLASS HIGHLVL POSITI
GRANT USAGE ON WORKLOAD WI_ADMIN TO PUBLIC;
CREATE WORKLOAD WI_PROD APPLNAME ('dss.exe') DISABLE SERVICE CLASS HIGHLVL POSITION A
GRANT USAGE ON WORKLOAD WI_PROD TO PUBLIC;
ALTER SERVICE CLASS HIGHLVL ENABLE;
ALTER WORKLOAD WI_BATCH ENABLE;
ALTER WORKLOAD WI_ADMIN ENABLE;
ALTER WORKLOAD WI_PROD ENABLE;
```

Figure 8-25 Generated code for DB2

Figure 8-26 on page 216 shows the AIX WLM class generated under WLMDEMO_BY_OBJ.osclasses.

Note: Any operating system WLM entities that you set up using Design Studio will *not* be automatically created on the target AIX machine. Users have to manually copy generated operating system code to the 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.



```
#
# mkclass commands for scheme: WLMDEMO_BY_OBJ
#
mkclass -a authuser=root -a adminuser=root DB2SC
```

Figure 8-26 Code generation for operating system WLM

Create limits for database activities

In this section, we demonstrate how to use the Create limits for database activities method to create the WLM scheme to manage poor database activities that degrade overall performance.

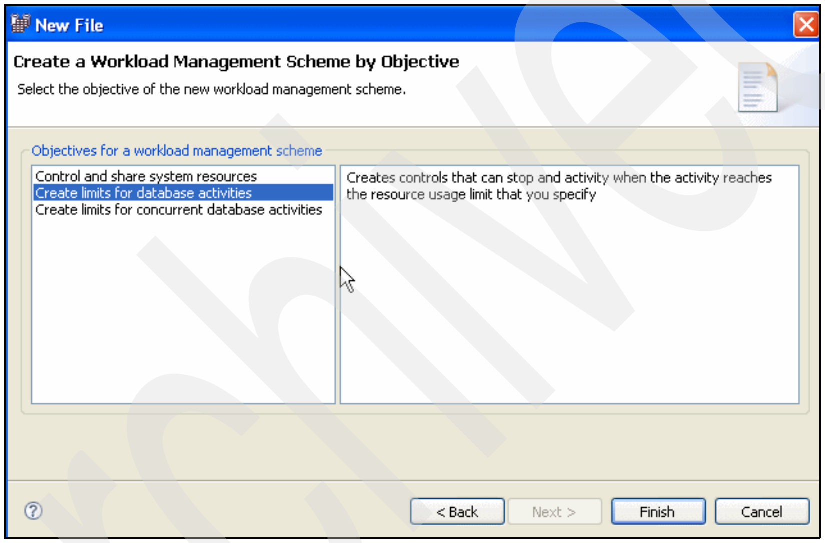
For example, a database system has reports run by a Sales department which usually run in five to ten minutes. On one occasion, a weekly sales report ran for six hours. Your reporting application has a built-in timeout limit. The application waits up to ten minutes for a query to return data, and then 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 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 the Create a Workload Management Scheme by Objective panel, select **Creating limits for database activities** (see Figure 8-27) and click **Finish**.

Note: Design Studio provides you with an alternate way to reach the Create limits for database activities screen. First, select the Database management scheme you want to work on and then go to **Workload Management -> Create Limits for Activities**.



Create a Workload Management Scheme by Objective

Select the objective of the new workload management scheme.

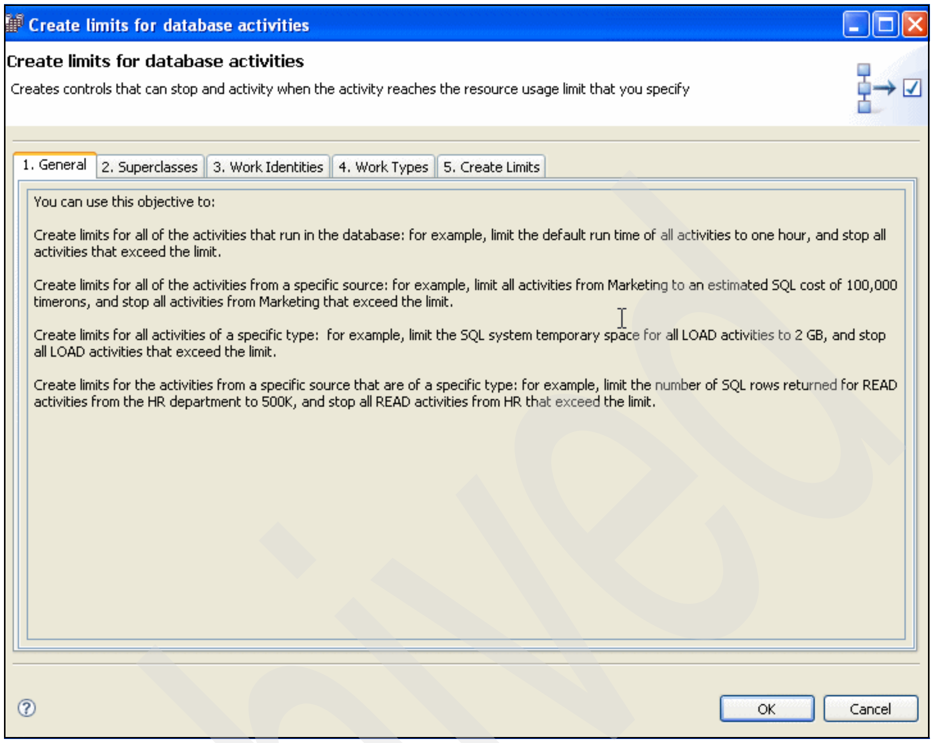
Objectives for a workload management scheme

Control and share system resources
Create limits for database activities
Create limits for concurrent database activities

Creates controls that can stop and activity when the activity reaches the resource usage limit that you specify

Figure 8-27 Create limits for database activities selection screen

The Create limits for database activities screen will then be presented, showing five tabs; see Figure 8-28 on page 218.


Create limits for database activities

Create limits for database activities

Creates controls that can stop and activity when the activity reaches the resource usage limit that you specify

You can use this objective to:

Create limits for all of the activities that run in the database: for example, limit the default run time of all activities to one hour, and stop all activities that exceed the limit.

Create limits for all of the activities from a specific source: for example, limit all activities from Marketing to an estimated SQL cost of 100,000 timerons, and stop all activities from Marketing that exceed the limit.

Create limits for all activities of a specific type: for example, limit the SQL system temporary space for all LOAD activities to 2 GB, and stop all LOAD activities that exceed the limit.

Create limits for the activities from a specific source that are of a specific type: for example, limit the number of SQL rows returned for READ activities from the HR department to 500K, and stop all READ activities from HR that exceed the limit.

Figure 8-28 Create limits for database activities - General tab

We explain the tabs here:

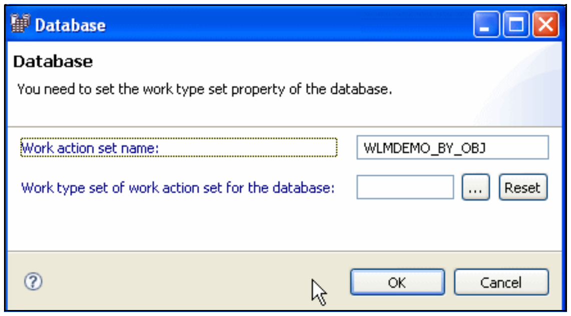
1. General

   This is an informational tab. Select the **Superclasses** tab to continue.

2. Superclasses

   Figure 8-29 shows the Superclasses tab with three default superclasses: SYSDEFAULTUSERCLASS, SYSDEFAULTSYSTEMCLASS, and SYSDEFAULTMAINTENANCELASS. Use this tab to add new superclasses, or to delete or modify existing superclasses.

   To add a superclass, click **Add** and the superclass property view is presented (refer to Figure 8-14 on page 206). Here you enter the name of the superclass to be created, the agent and prefetch priority, and the work action set name, if already known. These fields match AGENT PRIORITY, PREFETCH PRIORITY in the DB2 CREATE SERVICE CLASS statement.



Database

You need to set the work type set property of the database.

Work action set name: WLMDEMO_BY_OBJ

Work type set of work action set for the database:

Figure 8-29 Creating superclass in Create limits for database activities

3. Work Identities

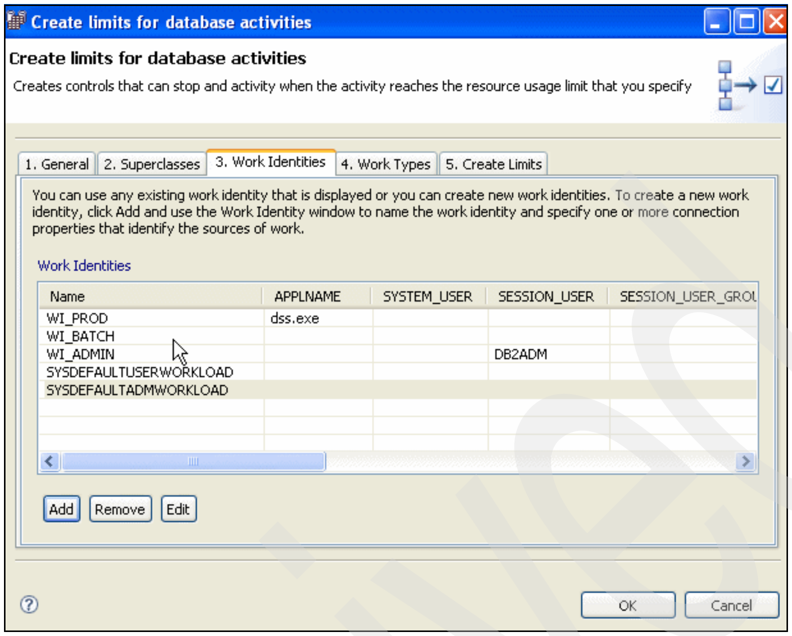
   You can use this tab to add a new DB2 WLM work identity, or to delete or modify an existing work identity.

   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 8-12 on page 203. You can identify the connection attributes to be used in your workload. To view the description of a field, left-click 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 workload execution authority.

   In our case, we created a workload WI_PROD to manage application dss.exe; see Figure 8-30. The workload can be used by everyone (public). We selected the **Work Types** tab to proceed.


Create limits for database activities

Create limits for database activities

Creates controls that can stop and activity when the activity reaches the resource usage limit that you specify

You can use any existing work identity that is displayed or you can create new work identities. To create a new work identity, click Add and use the Work Identity window to name the work identity and specify one or more connection properties that identify the sources of work.

Work Identities

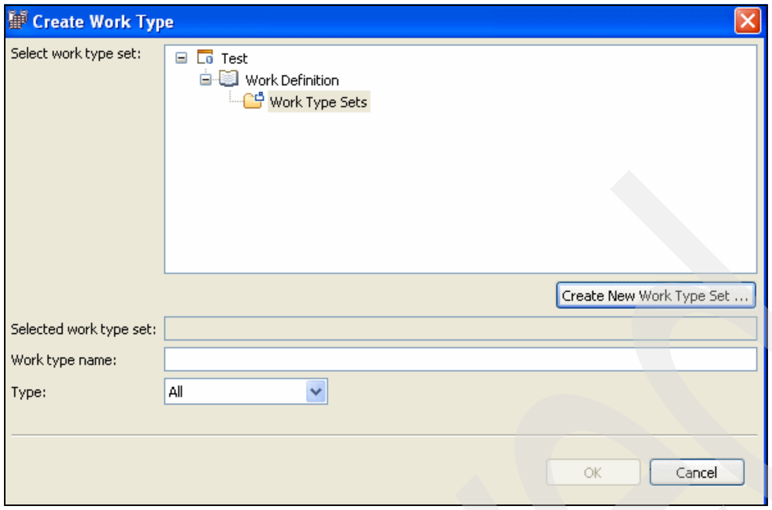
Name	APPLNAME	SYSTEM_USER	SESSION_USER	SESSION_USER_GROU
WI_PROD	dss.exe			
WI_BATCH				
WI_ADMIN			DB2ADM	
SYSDEFAULTUSERWORKLOAD				
SYSDEFAULTADMWORKLOAD				

Figure 8-30 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 8-31 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...**.


Create Work Type

Select work type set:

- Test
  - Work Definition
    - Work Type Sets

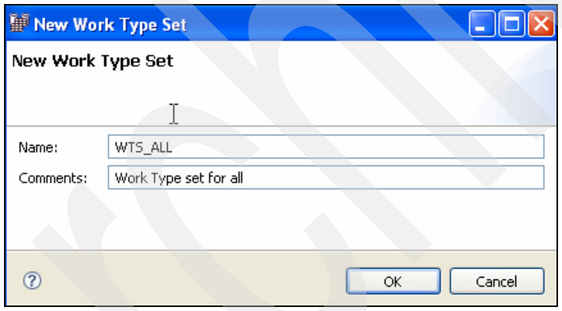
Selected work type set:

Work type name:

Type: All

Figure 8-31   Create work type main screen

Figure 8-32 shows the **Create New Work Type Set** property view. In our example, we created a Work type set WTS_ALL.


New Work Type Set

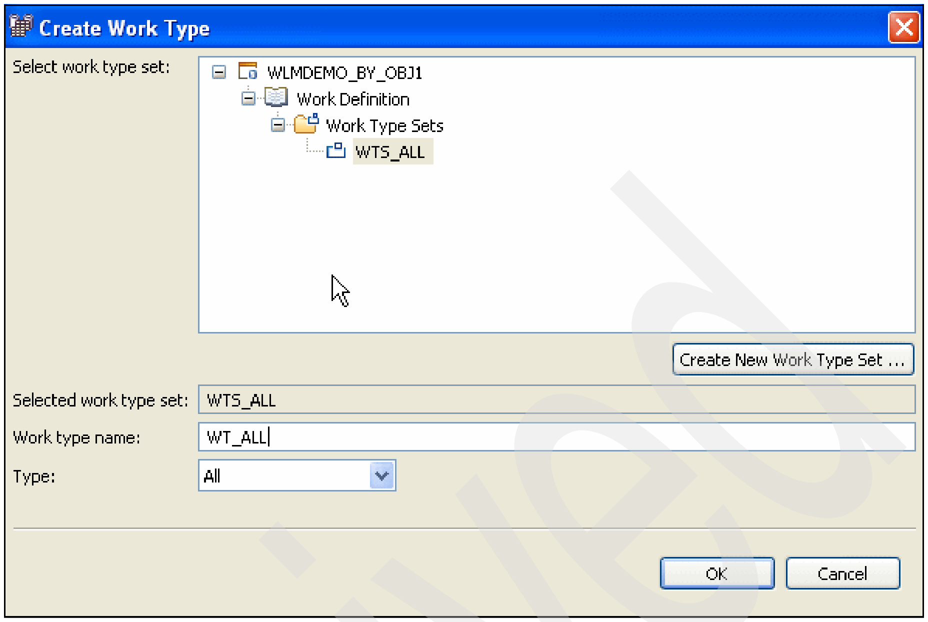
New Work Type Set

Name: WTS_ALL

Comments: Work Type set for all

Figure 8-32   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 8-33 on page 222.



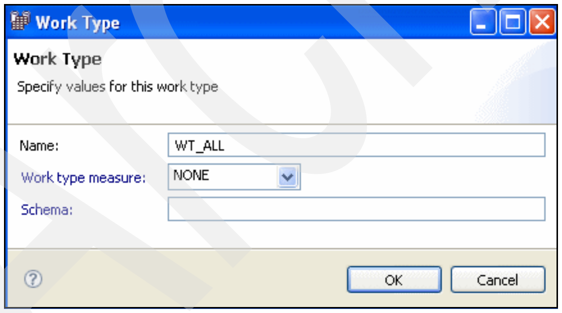
Selected work type set: WTS_ALL

Work type name: WT_ALL

Type: All

Figure 8-33 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 wanted to apply WT_ALL regardless of the measurement, so we choose **NONE** as shown in Figure 8-34.



Name: WT_ALL

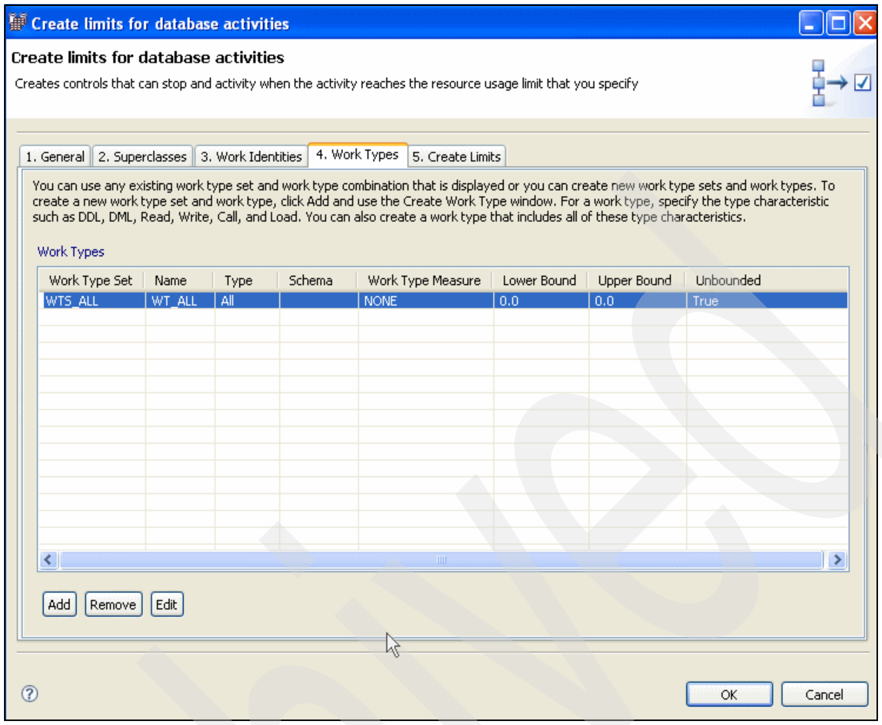
Work type measure: NONE

Schema:

Figure 8-34 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 8-35 on page 223 shows one work type set is created. You can create more type sets by following the same process. After completion, select the **Create Limits** tab to continue.


Create limits for database activities

Create limits for database activities

Creates controls that can stop and activity when the activity reaches the resource usage limit that you specify

1. General | 2. Superclasses | 3. Work Identities | 4. Work Types | 5. Create Limits

You can use any existing work type set and work type combination that is displayed or you can create new work type sets and work types. To create a new work type set and work type, click Add and use the Create Work Type window. For a work type, specify the type characteristic such as DDL, DML, Read, Write, Call, and Load. You can also create a work type that includes all of these type characteristics.

Work Types

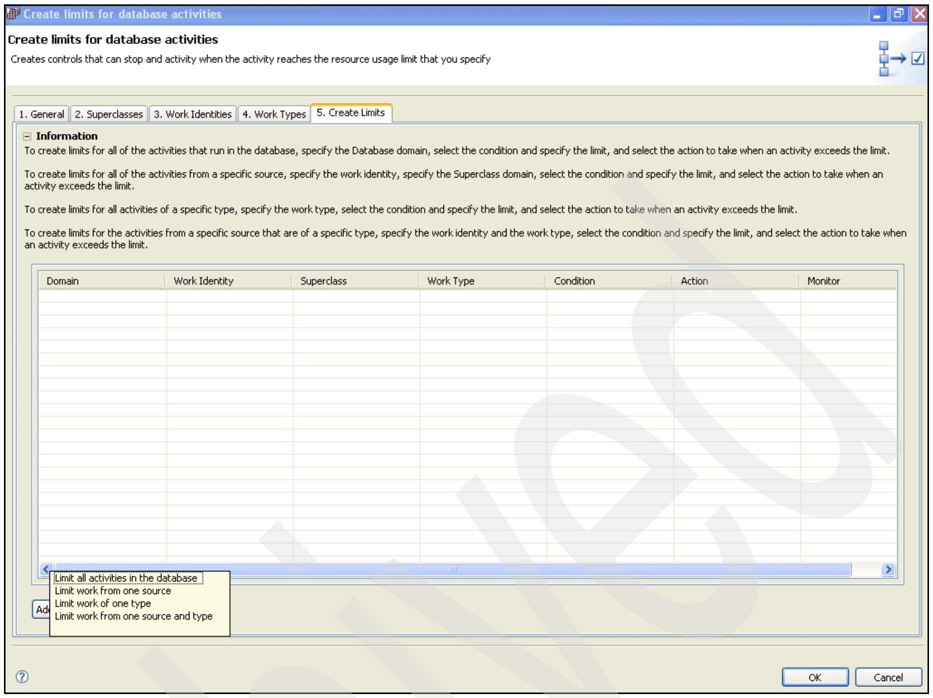
Work Type Set	Name	Type	Schema	Work Type Measure	Lower Bound	Upper Bound	Unbounded
WTS_ALL	WT_ALL	All		NONE	0.0	0.0	True

Figure 8-35 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 Limits

   In the Create Limits tab, click **Add** and then select the type of limit that you want to create; see Figure 8-36.



Create limits for database activities

Create limits for database activities

Creates controls that can stop and activity when the activity reaches the resource usage limit that you specify

1. General | 2. Superclasses | 3. Work Identities | 4. Work Types | 5. Create Limits

Information

To create limits for all of the activities that run in the database, specify the Database domain, select the condition and specify the limit, and select the action to take when an activity exceeds the limit.

To create limits for all of the activities from a specific source, specify the work identity, specify the Superclass domain, select the condition and specify the limit, and select the action to take when an activity exceeds the limit.

To create limits for all activities of a specific type, specify the work type, select the condition and specify the limit, and select the action to take when an activity exceeds the limit.

To create limits for the activities from a specific source that are of a specific type, specify the work identity and the work type, select the condition and specify the limit, and select the action to take when an activity exceeds the limit.

Domain	Work Identity	Superclass	Work Type	Condition	Action	Monitor

Limit all activities in the database
Limit work from one source
Limit work of one type
Limit work from one source and type
Ad
OK
Cancel

Figure 8-36 Creating limits for activities

There are four options:

– Limit all activities in the database

  You can use this option when a limit has to be applied for a 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

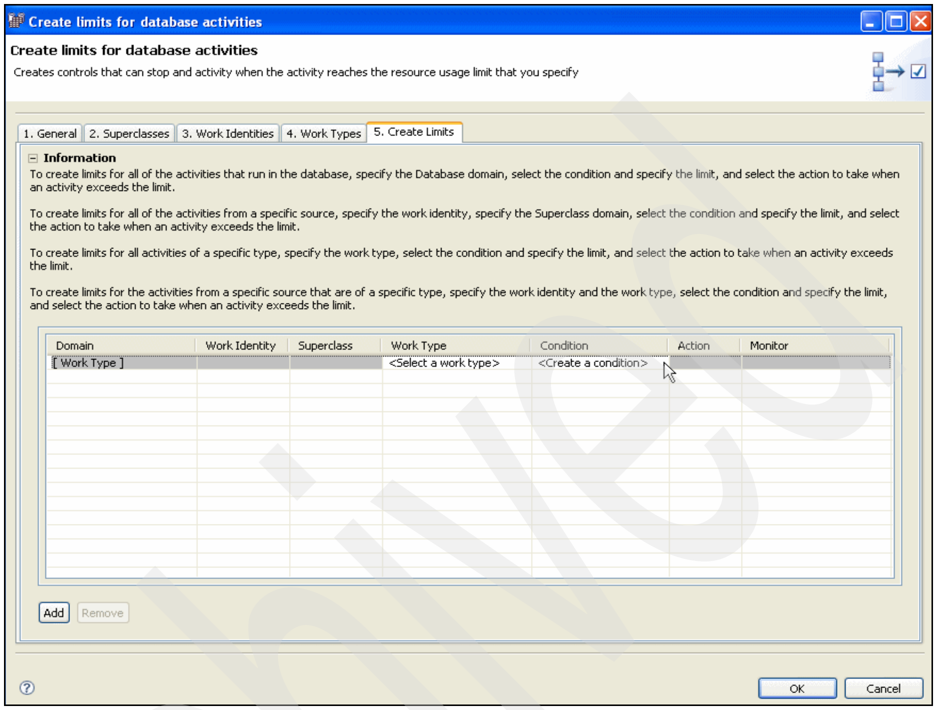
  You can use this option when limit has to be associated with only one work type.

– Limit work of one source and type

  You can use this option to limit work from a specific workload, of specific types (for example, READ, WRITE, and so on).

For each option selected, Design Studio adds one entry with the required fields “opened” (not greyed out). In our example, we selected **Limit work of**

one type to control problem queries on the database. Figure 8-37 shows that for domain Work Type, the required fields are Work Type and Condition.



Create limits for database activities

Create limits for database activities

Creates controls that can stop and activity when the activity reaches the resource usage limit that you specify

1. General | 2. Superclasses | 3. Work Identities | 4. Work Types | 5. Create Limits

Information

To create limits for all of the activities that run in the database, specify the Database domain, select the condition and specify the limit, and select the action to take when an activity exceeds the limit.

To create limits for all of the activities from a specific source, specify the work identity, specify the Superclass domain, select the condition and specify the limit, and select the action to take when an activity exceeds the limit.

To create limits for all activities of a specific type, specify the work type, select the condition and specify the limit, and select the action to take when an activity exceeds the limit.

To create limits for the activities from a specific source that are of a specific type, specify the work identity and the work type, select the condition and specify the limit, and select the action to take when an activity exceeds the limit.

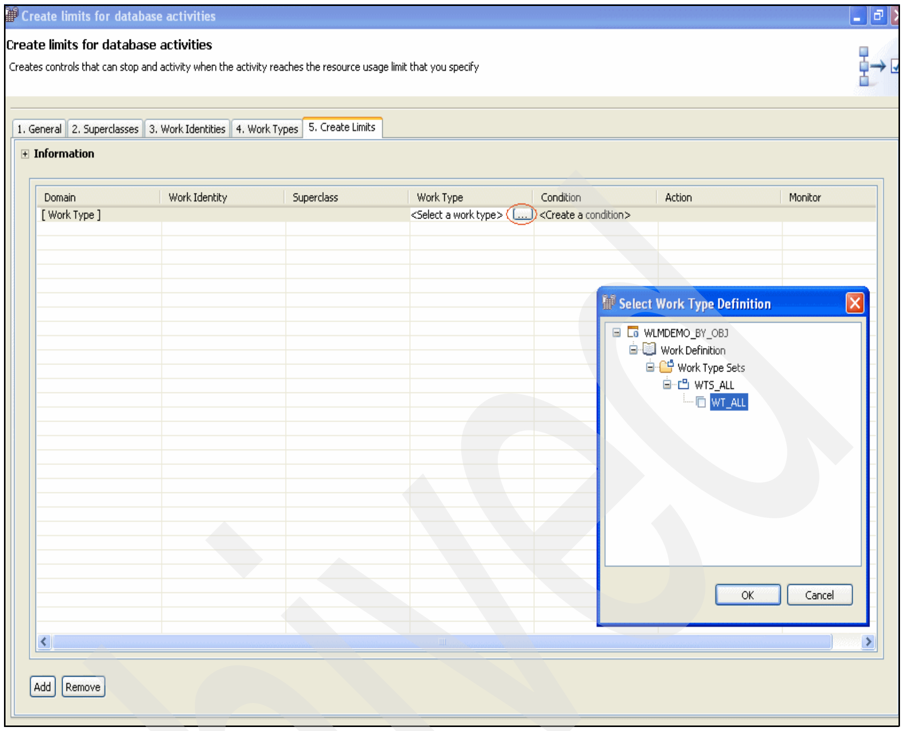
Domain	Work Identity	Superclass	Work Type	Condition	Action	Monitor
[Work Type]			<Select a work type>	<Create a condition>		

Add | Remove

OK | Cancel

Figure 8-37 Creating limits using Limit work of one type

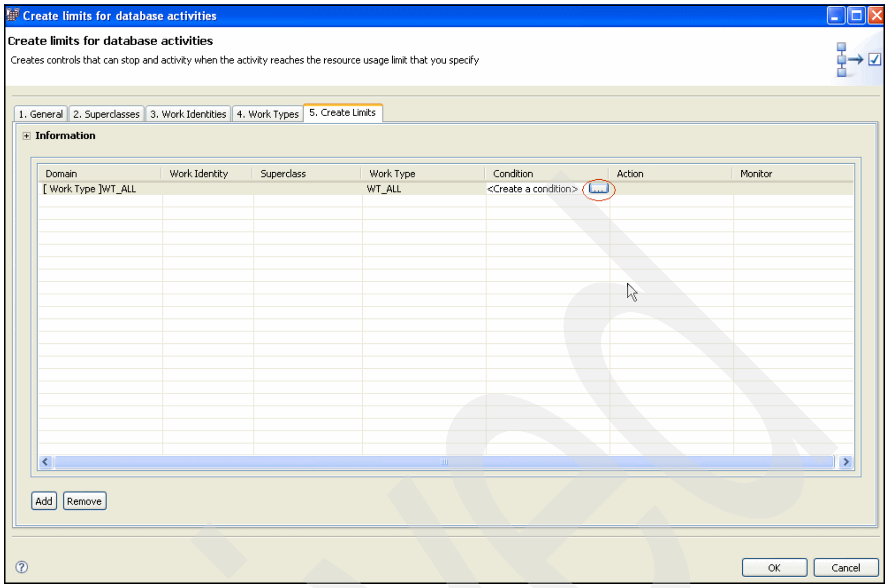
To specify the work type that you want to limit, click **<select a work type>** and a browse button shows in the field. Click the browse button to select the list of work type sets you can choose from; see Figure 8-38.


Create limits for database activities
Create limits for database activities
Creates controls that can stop and activity when the activity reaches the resource usage limit that you specify
1. General
2. Superclasses
3. Work Identities
4. Work Types
5. Create Limits
Information
Select Work Type Definition
WLMDEMO_BY_OBJ
Work Definition
Work Type Sets
WTS_ALL
WT_ALL
OK
Cancel
Add
Remove

Domain	Work Identity	Superclass	Work Type	Condition	Action	Monitor
[Work Type]			<Select a work type> ...	<Create a condition>		

Figure 8-38 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, and a browse button shows in the field; see Figure 8-39.



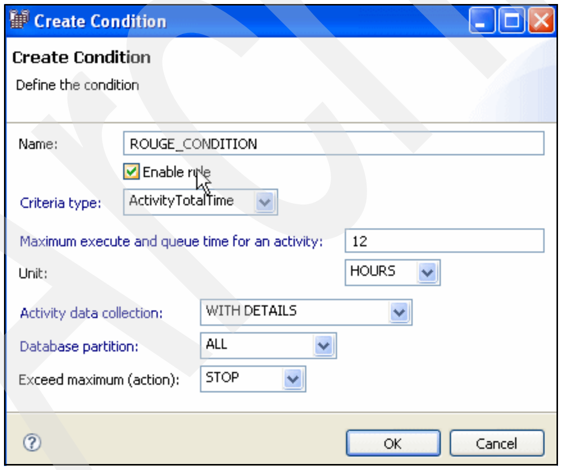
Create limits for database activities

Creates controls that can stop and activity when the activity reaches the resource usage limit that you specify

Domain	Work Identity	Superclass	Work Type	Condition	Action	Monitor
[Work Type]WT_ALL			WT_ALL	<Create a condition>		

Figure 8-39   Creating limits associated with new condition

Click the 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 8-40 on page 227.



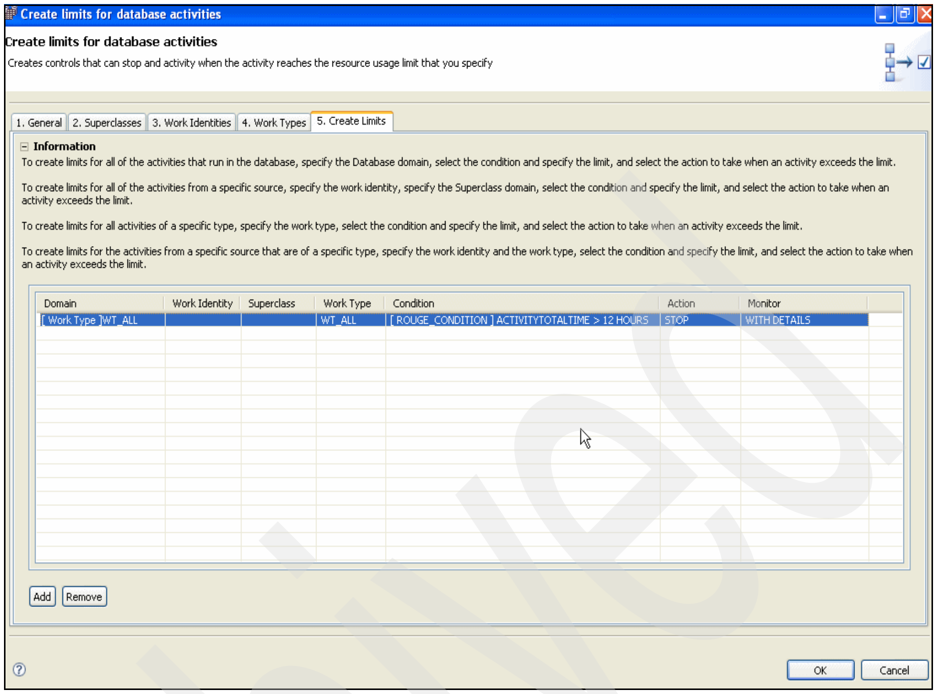
Create Condition

Define the condition

Name:	ROUGE_CONDITION
	✓ Enable rule
Criteria type:	ActivityTotalTime
Maximum execute and queue time for an activity:	12
Unit:	HOURS
Activity data collection:	WITH DETAILS
Database partition:	ALL
Exceed maximum (action):	STOP

Figure 8-40   Limit work of one type - specify condition

Figure 8-41 shows that one limit is created.



Create limits for database activities

Creates controls that can stop and activity when the activity reaches the resource usage limit that you specify

Information

To create limits for all of the activities that run in the database, specify the Database domain, select the condition and specify the limit, and select the action to take when an activity exceeds the limit.

To create limits for all of the activities from a specific source, specify the work identity, specify the Superclass domain, select the condition and specify the limit, and select the action to take when an activity exceeds the limit.

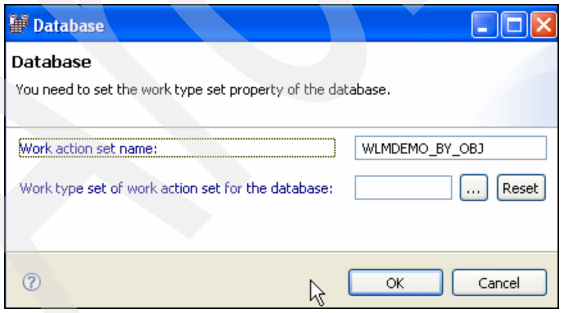
To create limits for all activities of a specific type, specify the work type, select the condition and specify the limit, and select the action to take when an activity exceeds the limit.

To create limits for the activities from a specific source that are of a specific type, specify the work identity and the work type, select the condition and specify the limit, and select the action to take when an activity exceeds the limit.

Domain	Work Identity	Superclass	Work Type	Condition	Action	Monitor
[Work Type]WT_ALL			WT_ALL	[ROUGE_CONDITION] ACTIVITYTOTALTIME > 12 HOURS	STOP	WITH DETAILS

Figure 8-41 One limit created

You can continue adding limits. When you are finished, click **OK** and the Database panel will be presented, allowing you to associate the limits to a work action set. Click the browse button to specify the work action set; see Figure 8-42 on page 228.



Database

You need to set the work type set property of the database.

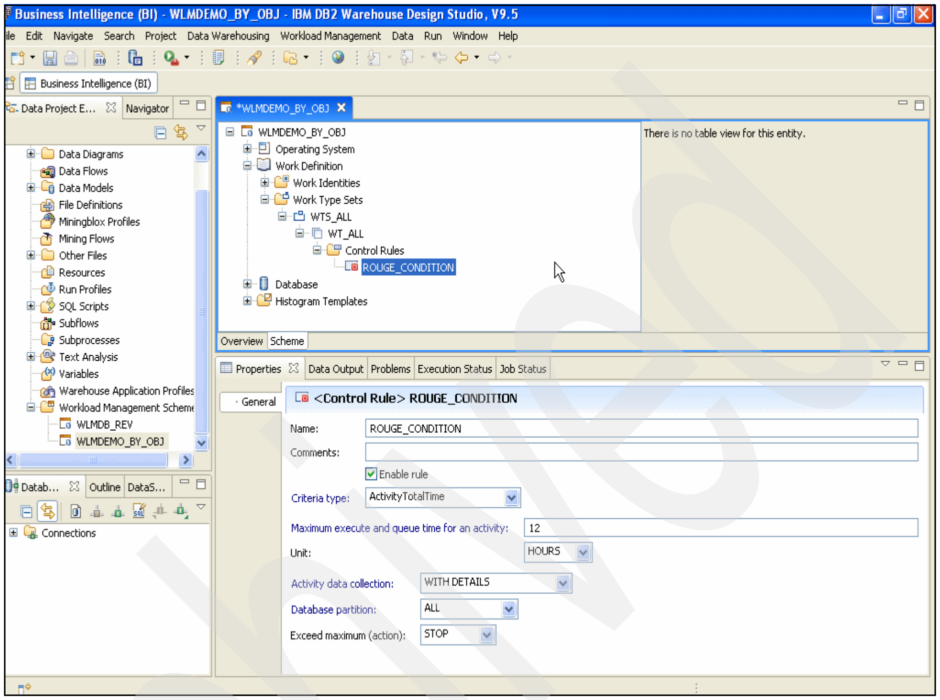
Work action set name: WLMDEMO_BY_OBJ

Work type set of work action set for the database:

Figure 8-42 Work type set property of the database

After associating the work action, click **OK**. Design Studio creates a WLM Scheme and returns you to the default Business Intelligence (BI) perspective.

In the BI perspective, Design Studio focuses you on the Editors panel and presents the overview of the current project created. Figure 8-43 shows the project we created, WLMDEMO_BY_OBJ, and its control rules.



There is no table view for this entity.

<Control Rule> ROUGE_CONDITION

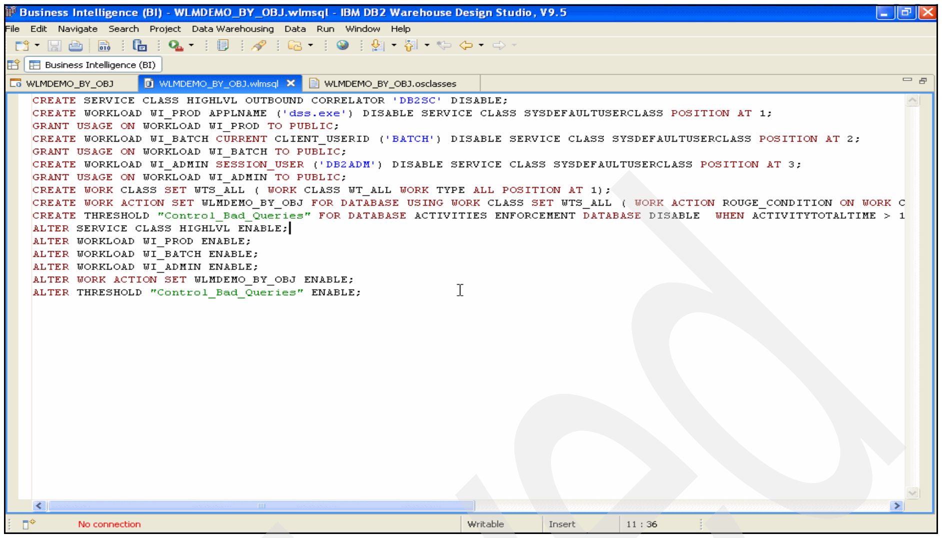
Name: ROUGE_CONDITION
Comments:
[✓] Enable rule
Criteria type: ActivityTotalTime
Maximum execute and queue time for an activity: 12
Unit: HOURS
Activity data collection: WITH DETAILS
Database partition: ALL
Exceed maximum (action): STOP

Figure 8-43 Control Rule - WLM scheme view

Validating the WLM scheme and generating code

After 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 by using the Generate Code feature. For the details, refer to “Validating” on page 213 and “Generating code” on page 214.

Figure 8-44 on page 230 shows the generated WLMDEMO_BY_OBJ.wlmsql for creating limits for database activities.



```
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 ('DB2ADM') 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;
```

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
No connection
Writable
Insert
11 : 36

Figure 8-44 Create limits for database activities - generated code

The Design Studio guided configuration Create limits on database activities supports the following solution templates, which map to the DB2 CREATE THRESHOLD statement.

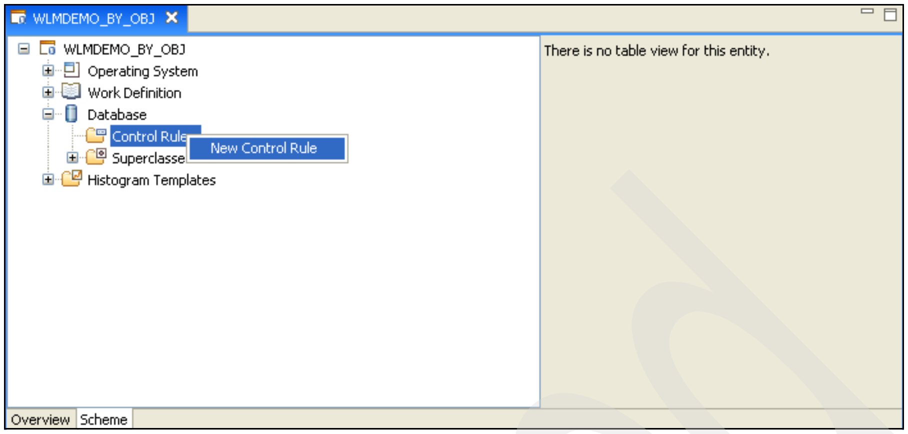
- Create control rule on database
- Create control rule on superclass
- Create control rule for a work type (WHEN)
- Work type on a superclass that maps to a subclass with a control rule

Design Studio provides 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 the Scheme tab in the BI perspective. Control rules can be added or modified for work type sets under the Work Definition, or for a database. In this section, we demonstrate how to add a new database control rule.

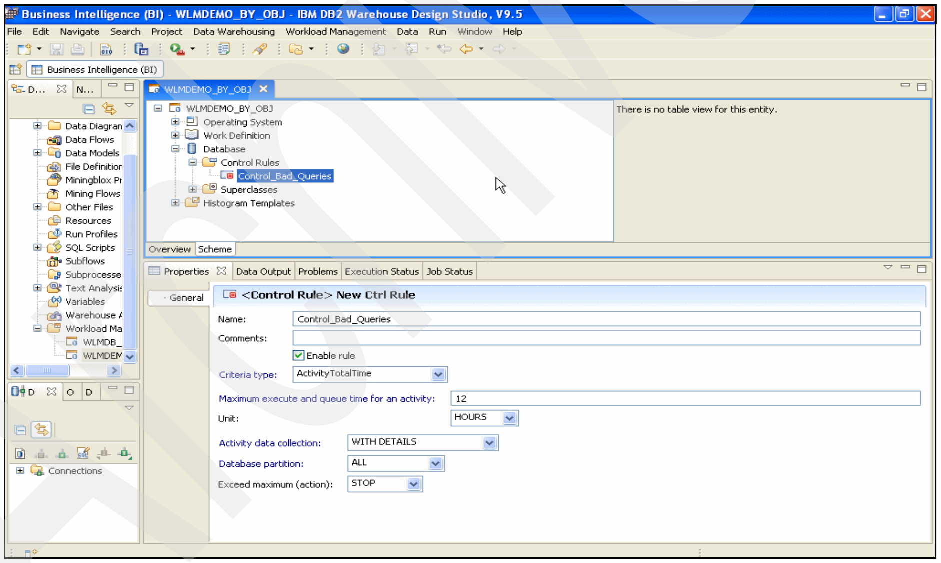
To create a New Control Rule, expand the WLM scheme and Database, then right-click **Control rules**; see Figure 8-45.



There is no table view for this entity.

Figure 8-45 Control Rule Creation using a tree view

You can complete the fields in the Properties view for the new control rule; see Figure 8-46.



There is no table view for this entity.

<Control Rule> New Ctrl Rule

Name:	Control_Bad_Queries
Comments:	
	✓ Enable rule
Criteria type:	ActivityTotalTime
Maximum execute and queue time for an activity:	12
Unit:	HOURS
Activity data collection:	WITH DETAILS
Database partition:	ALL
Exceed maximum (action):	STOP

Figure 8-46 Control Rule for Database

When you create the Control Rule for Database using Design Studio, it generates code that 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 the control rule based on activity type
  - Stop activity that consumes excessive temporary space.
  - Allow one activity to take a 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

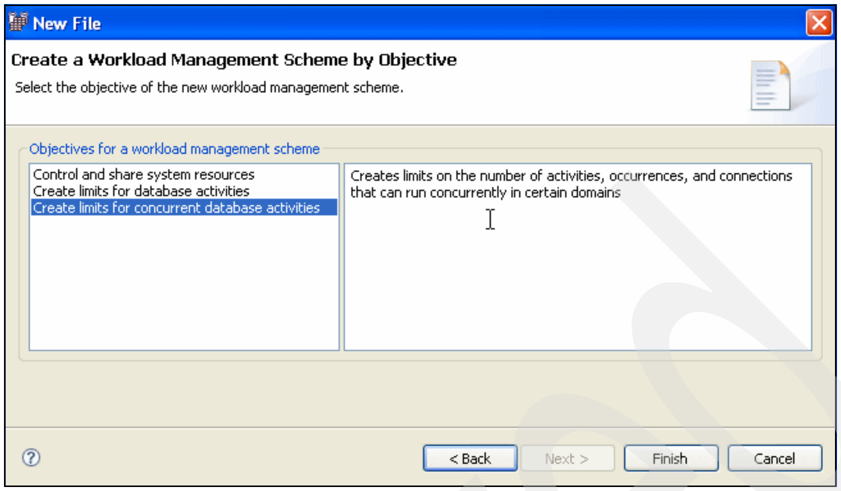
You can use the Create limits for concurrent activities method to create and enforce concurrency limits on database activities. You also specify whether to stop activities that cause the concurrency limits to be exceeded, or allow them to continue to execute.

For example, suppose a database administrator noticed that there were many simultaneous activities happening on a specific work identity, and wanted 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

This would be accomplished by selecting the Create limits for concurrent activities from WLM Scheme by Objectives screen, expanding the project tree view (WLMDB, in our example), right-clicking **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 (shown in Figure 8-47), select **Create limits for concurrent database activities** and click **Finish**.



Create a Workload Management Scheme by Objective

Select the objective of the new workload management scheme.

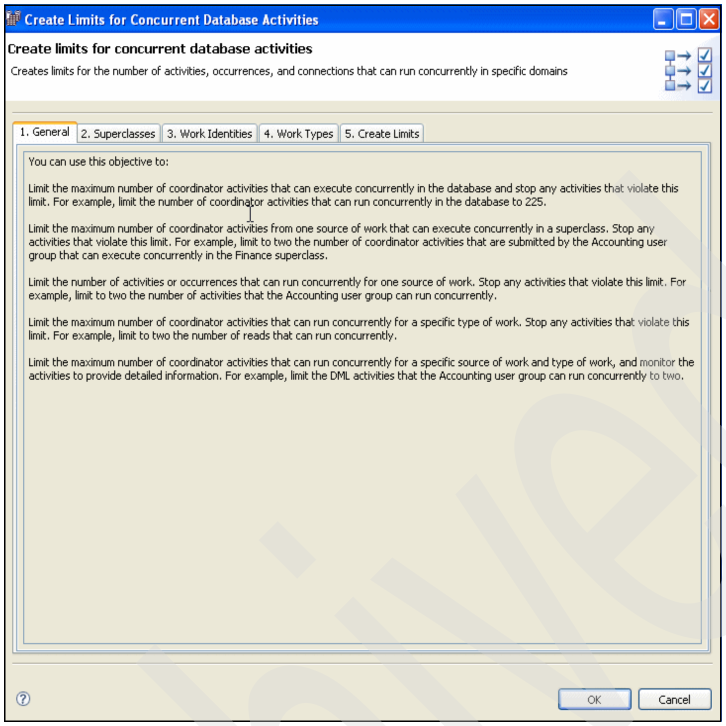
Objectives for a workload management scheme

Control and share system resources
Create limits for database activities
Create limits for concurrent database activities

Creates limits on the number of activities, occurrences, and connections that can run concurrently in certain domains

Figure 8-47 Create limits for concurrent database activities

The Create limits for the concurrent database activities main screen contains five tabs: General, Superclasses, Work Identities, Work Types and Create Limits; see Figure 8-48.



Create limits for concurrent database activities

Creates limits for the number of activities, occurrences, and connections that can run concurrently in specific domains

You can use this objective to:

Limit the maximum number of coordinator activities that can execute concurrently in the database and stop any activities that violate this limit. For example, limit the number of coordinator activities that can run concurrently in the database to 225.

Limit the maximum number of coordinator activities from one source of work that can execute concurrently in a superclass. Stop any activities that violate this limit. For example, limit to two the number of coordinator activities that are submitted by the Accounting user group that can execute concurrently in the Finance superclass.

Limit the number of activities or occurrences that can run concurrently for one source of work. Stop any activities that violate this limit. For example, limit to two the number of activities that the Accounting user group can run concurrently.

Limit the maximum number of coordinator activities that can run concurrently for a specific type of work. Stop any activities that violate this limit. For example, limit to two the number of reads that can run concurrently.

Limit the maximum number of coordinator activities that can run concurrently for a specific source of work and type of work, and monitor the activities to provide detailed information. For example, limit the DML activities that the Accounting user group can run concurrently to two.

Figure 8-48 Create limits for concurrent database activities - Main screen

- General

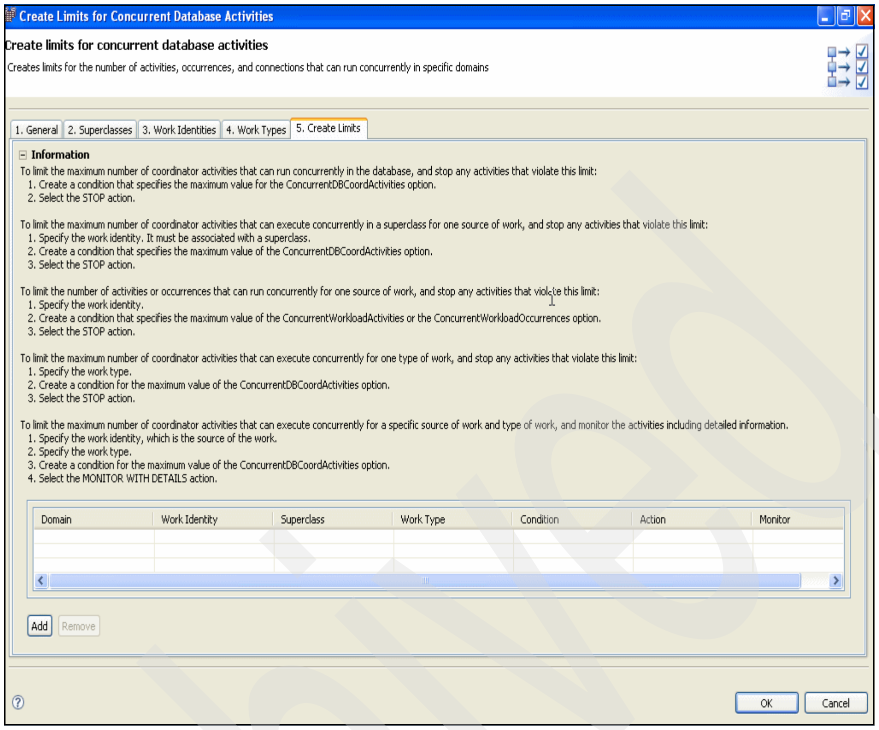
  This is an informational tab. Select the **Superclasses** tab to continue.

- Superclasses, Work Identities, and Work Type sets

  Use the same process described in “Controlling and sharing system resources” on page 200, to create superclasses, work identities, and work type sets.

- Create Limits

  Use this tab (shown in Figure 8-49 on page 235) to create limits for concurrent activities.


Create Limits for Concurrent Database Activities

Create limits for concurrent database activities

Creates limits for the number of activities, occurrences, and connections that can run concurrently in specific domains

1. General | 2. Superclasses | 3. Work Identities | 4. Work Types | 5. Create Limits

Information

To limit the maximum number of coordinator activities that can run concurrently in the database, and stop any activities that violate this limit:
1. Create a condition that specifies the maximum value for the ConcurrentDBCoordActivities option.
2. Select the STOP action.

To limit the maximum number of coordinator activities that can execute concurrently in a superclass for one source of work, and stop any activities that violate this limit:
1. Specify the work identity. It must be associated with a superclass.
2. Create a condition that specifies the maximum value of the ConcurrentDBCoordActivities option.
3. Select the STOP action.

To limit the number of activities or occurrences that can run concurrently for one source of work, and stop any activities that violate this limit:
1. Specify the work identity.
2. Create a condition that specifies the maximum value of the ConcurrentWorkloadActivities or the ConcurrentWorkloadOccurrences option.
3. Select the STOP action.

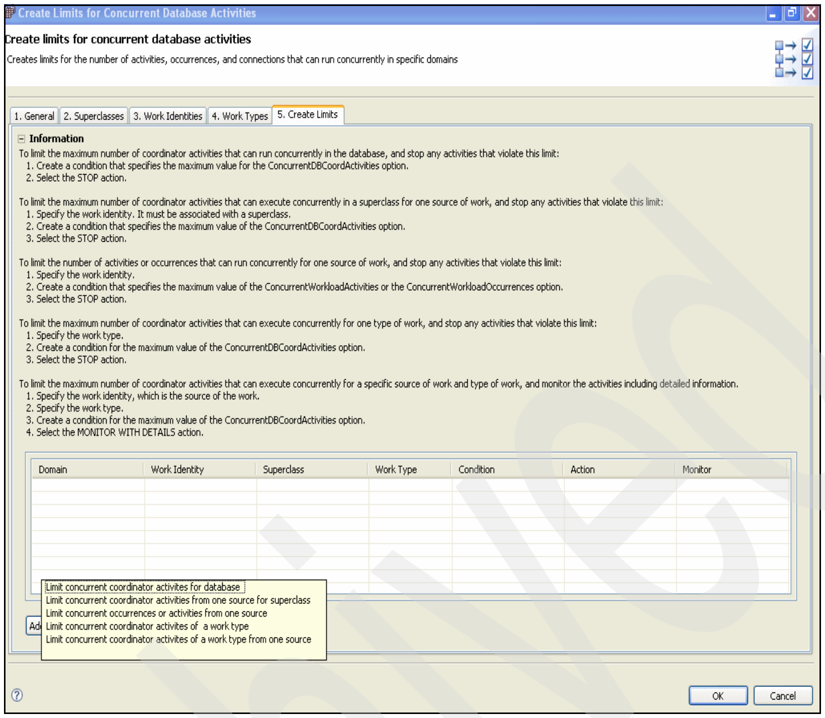
To limit the maximum number of coordinator activities that can execute concurrently for one type of work, and stop any activities that violate this limit:
1. Specify the work type.
2. Create a condition for the maximum value of the ConcurrentDBCoordActivities option.
3. Select the STOP action.

To limit the maximum number of coordinator activities that can execute concurrently for a specific source of work and type of work, and monitor the activities including detailed information.
1. Specify the work identity, which is the source of the work.
2. Specify the work type.
3. Create a condition for the maximum value of the ConcurrentDBCoordActivities option.
4. Select the MONITOR WITH DETAILS action.

Domain	Work Identity	Superclass	Work Type	Condition	Action	Monitor

Figure 8-49 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 8-50 on page 236.



Create Limits for Concurrent Database Activities

Create limits for concurrent database activities

Creates limits for the number of activities, occurrences, and connections that can run concurrently in specific domains

1. General | 2. Superclasses | 3. Work Identities | 4. Work Types | 5. Create Limits

Information

To limit the maximum number of coordinator activities that can run concurrently in the database, and stop any activities that violate this limit:
1. Create a condition that specifies the maximum value for the ConcurrentDBCoordActivities option.
2. Select the STOP action.

To limit the maximum number of coordinator activities that can execute concurrently in a superclass for one source of work, and stop any activities that violate this limit:
1. Specify the work identity. It must be associated with a superclass.
2. Create a condition that specifies the maximum value of the ConcurrentDBCoordActivities option.
3. Select the STOP action.

To limit the number of activities or occurrences that can run concurrently for one source of work, and stop any activities that violate this limit:
1. Specify the work identity.
2. Create a condition that specifies the maximum value of the ConcurrentWorkloadActivities or the ConcurrentWorkloadOccurrences option.
3. Select the STOP action.

To limit the maximum number of coordinator activities that can execute concurrently for one type of work, and stop any activities that violate this limit:
1. Specify the work type.
2. Create a condition for the maximum value of the ConcurrentDBCoordActivities option.
3. Select the STOP action.

To limit the maximum number of coordinator activities that can execute concurrently for a specific source of work and type of work, and monitor the activities including detailed information.
1. Specify the work identity, which is the source of the work.
2. Specify the work type.
3. Create a condition for the maximum value of the ConcurrentDBCoordActivities option.
4. Select the MONITOR WITH DETAILS action.

Domain	Work Identity	Superclass	Work Type	Condition	Action	Monitor

Ad
Limit concurrent coordinator activites for database
Limit concurrent coordinator activities from one source for superclass
Limit concurrent occurrences or activities from one source
Limit concurrent coordinator activites of a work type
Limit concurrent coordinator activites of a work type from one source
OK
Cancel

Figure 8-50   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

  With this option you can:

  - Define the condition for a limit on the database domain.
  - Specify the type of action to be taken when the concurrency limit is exceeded.

– Limit concurrent coordinator activities from one source for superclass

  With 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 the type of action to be taken when the concurrency limit is exceeded.

– Limit concurrent occurrences or activities from one source

  With this option, you can:

  - 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

With this option, you can:

- Specify the work type that you want to limit.
- Define the condition for a limit on the work type domain.
- Specify the action to take when the concurrency limit is exceeded.

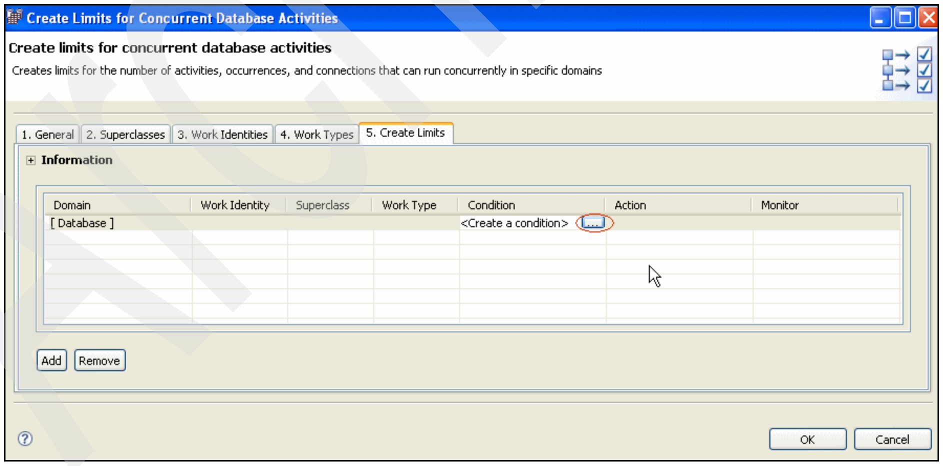
– Limit concurrent coordinator activities of a work type from one source

In this option, you can:

- 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 selected **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 the "opened" field; see Figure 8-51.


Create Limits for Concurrent Database Activities

Create limits for concurrent database activities

Creates limits for the number of activities, occurrences, and connections that can run concurrently in specific domains

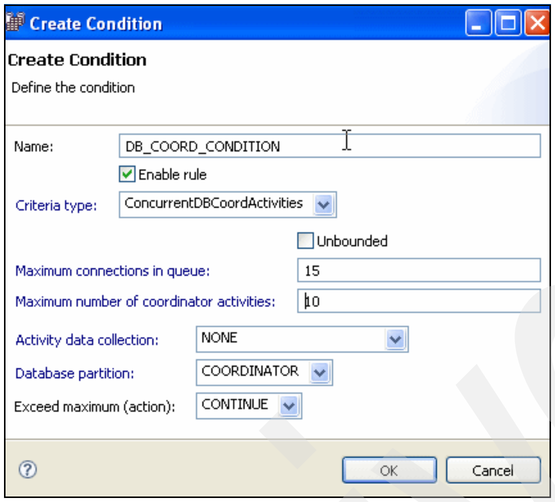
1. General | 2. Superclasses | 3. Work Identities | 4. Work Types | 5. Create Limits

Information

Domain	Work Identity	Superclass	Work Type	Condition	Action	Monitor
[Database]				<Create a condition> ...		

Figure 8-51 Create limits - Create a condition

In our example, the Create Condition panel presents where you can specify concurrency control configuration; see Figure 8-52. To see the description of a field, left-click the field name.


Create Condition

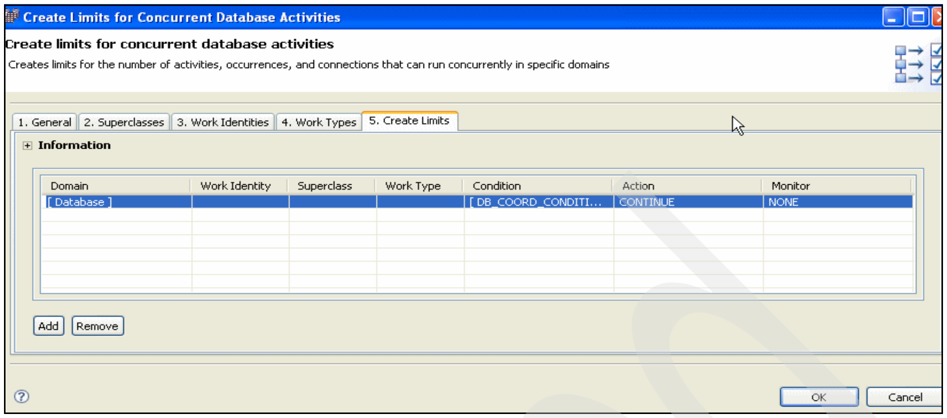
Create Condition
Define the condition

Name:	DB_COORD_CONDITION
	[✓] Enable rule
Criteria type:	ConcurrentDBCoordActivities
	[ ] Unbounded
Maximum connections in queue:	15
Maximum number of coordinator activities:	10
Activity data collection:	NONE
Database partition:	COORDINATOR
Exceed maximum (action):	CONTINUE

Figure 8-52 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 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 8-53.


Create Limits for Concurrent Database Activities

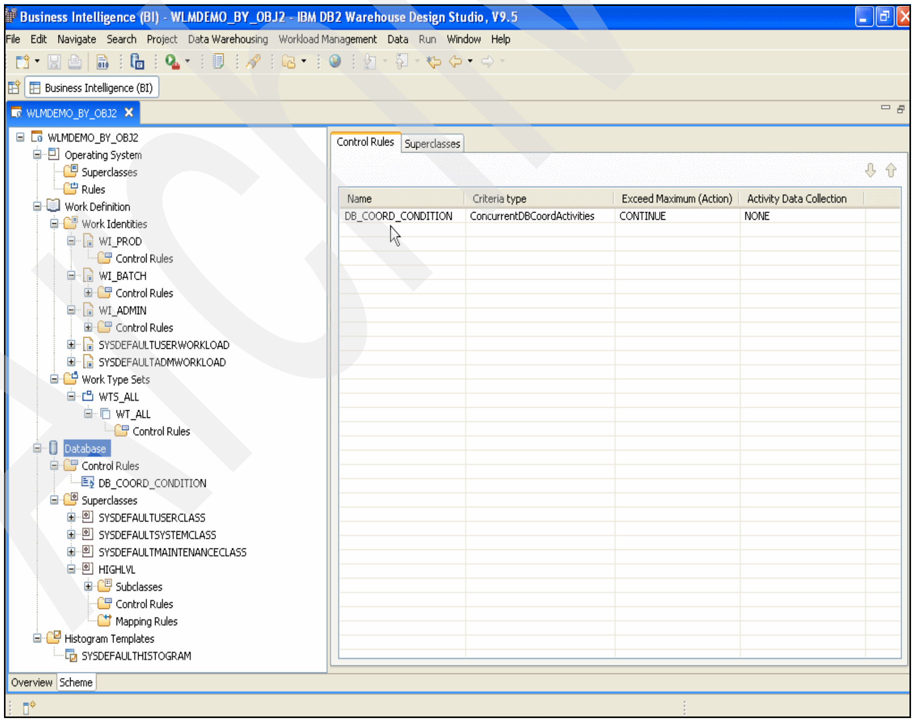
Create limits for concurrent database activities

Creates limits for the number of activities, occurrences, and connections that can run concurrently in specific domains

Domain	Work Identity	Superclass	Work Type	Condition	Action	Monitor
[Database]				[DB_COORD_CONDITI...	CONTINUE	NONE

Figure 8-53 Create limits for Concurrent database activities showing Control Rule

You can add additional control rules by using the **Add** button. After all the control rules are defined, click **OK**. Design Studio creates a WLM Scheme and returns you to the default Business Intelligence (BI) perspective. The BI Perspective is expanded to show the final output; see Figure 8-54.


Business Intelligence (BI) - WLMDEMO_BY_OBJ2 - IBM DB2 Warehouse Design Studio, V9.5

```
WLMDEMO_BY_OBJ2
  Operating System
    Superclasses
    Rules
  Work Definition
    Work Identities
      WI_PROD
        Control Rules
      WI_BATCH
        Control Rules
      WI_ADMIN
        Control Rules
      SYSDEFAULTUSERWORKLOAD
      SYSDEFAULTADMWORKLOAD
    Work Type Sets
      WTS_ALL
        WT_ALL
          Control Rules
  Database
    Control Rules
      DB_COORD_CONDITION
    Superclasses
      SYSDEFAULTUSERCLASS
      SYSDEFAULTSYSTEMCLASS
      SYSDEFAULTMAINTENANCECLASS
      HIGHLVL
        Subclasses
        Control Rules
        Mapping Rules
  Histogram Templates
    SYSDEFAULTHISTOGRAM
```

Name	Criteria type	Exceed Maximum (Action)	Activity Data Collection
DB_COORD_CONDITION	ConcurrentDBCoordActivities	CONTINUE	NONE

Figure 8-54 Control Rule to limit concurrent coordinator activities for database - tree view

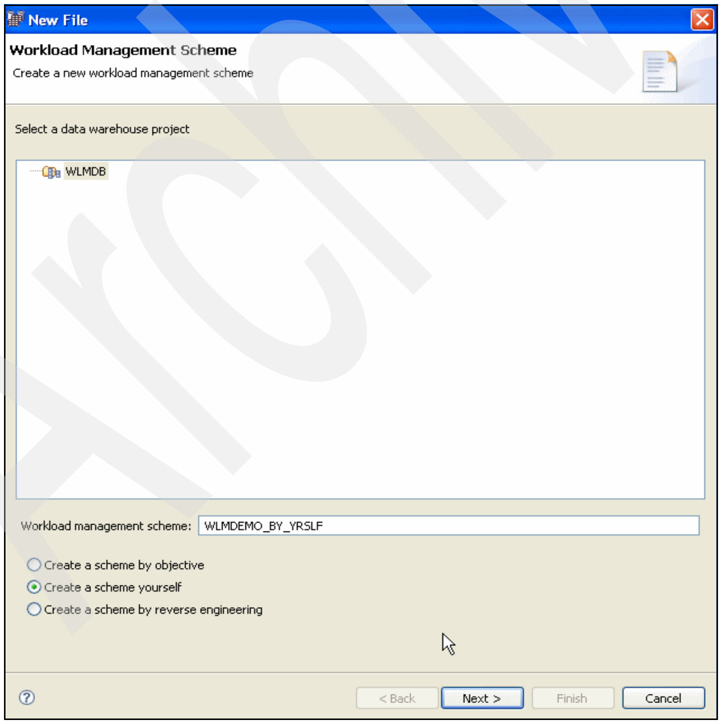
Validating WLM scheme and generating code

After 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 213 and “Generating code” on page 214.

8.3.2 Create workload scheme by yourself

You can use workload creating options to create a WLM scheme and work with entities in the scheme by creating everything yourself. 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 this way, select **Create a scheme yourself** and enter the WLM scheme name as shown in Figure 8-55. In this example, we created a WLM scheme WLMDEMO_BY_YRSLF.


New File
Workload Management Scheme
Create a new workload management scheme
Select a data warehouse project
WLMDB
< Back
Next >
Finish
Cancel

Workload management scheme: WLMDEMO_BY_YRSLF

Create a scheme by objective [✓ Create a scheme yourself] Create a scheme by reverse engineering

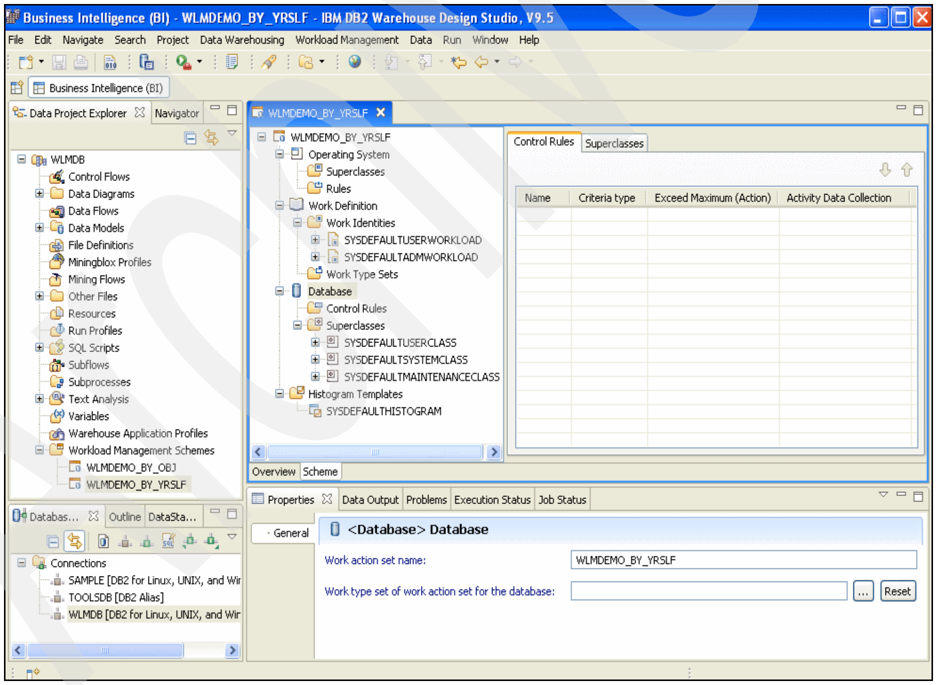
Figure 8-55 Create scheme yourself

Click **Next** and the Workload Management Scheme Options panel will be displayed, where you can define a work action set name. By default, it takes the WLM scheme name you provided as the work action set name. In our case, we kept the default work action set name and clicked **Finish**. The file name created is the scheme name with .wlms under the default workspace defined.

In addition to the default work identities, Design Studio also shows you these defaults:

- ► Default super classes (SYSDEFAULTSYSTEMCLASS, SYSDEFAULTUSERCLASS)
- ► Default subclass under every superclass (SYSDEFAULTSUBCLASS)
- ► Default histogram (SYSDEFAULTHISTOGRAM)

Design Studio creates the WLM scheme with only the default work identities, as shown in Figure 8-56 on page 241. Therefore, 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 by using the tree view.

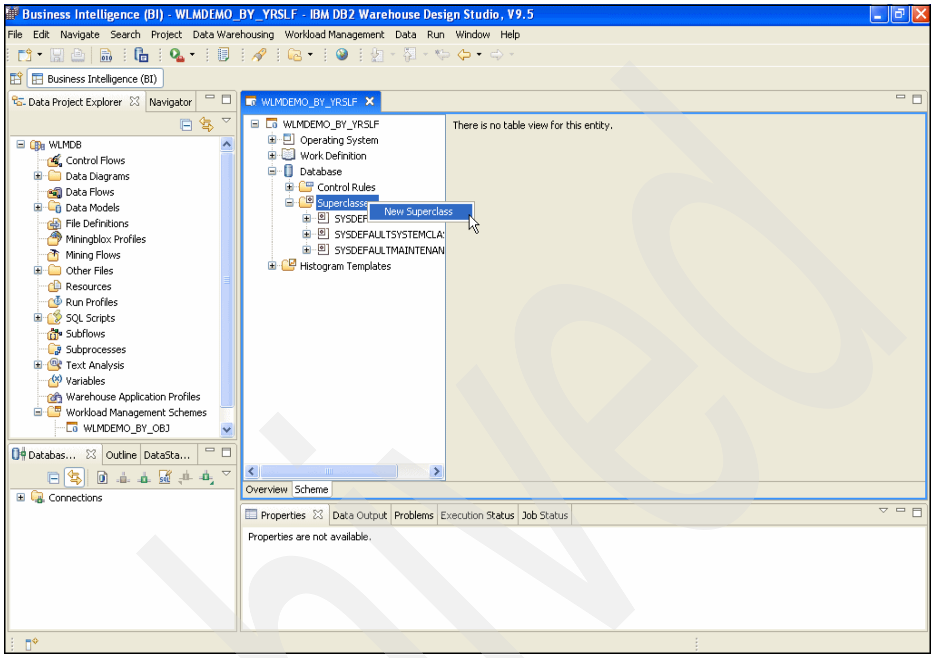


Work action set name: WLMDEMO_BY_YRSLF

Work type set of work action set for the database:

Figure 8-56 WLM Scheme created using Create by yourself option

To create a new entity, right-click 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 8-57.

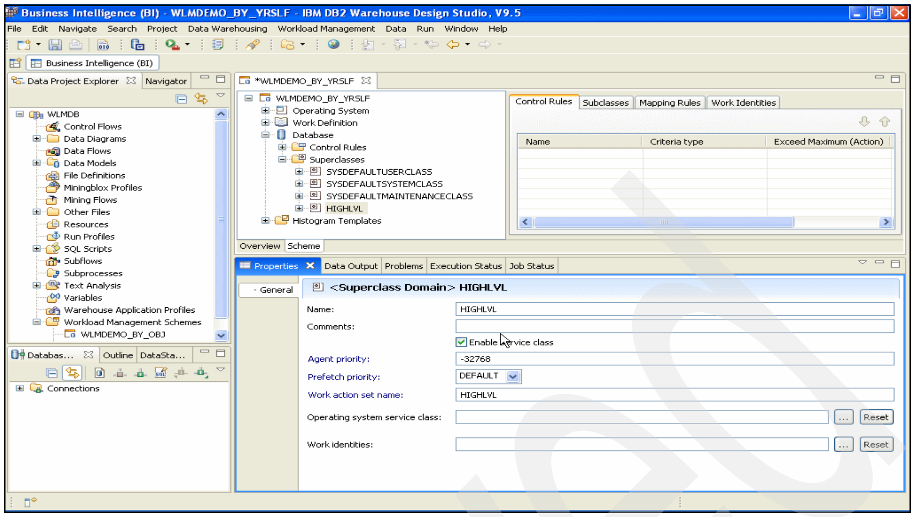


There is no table view for this entity.

Properties are not available.

Figure 8-57 Create Scheme by Yourself - Create New Superclass

You are required to enter a unique name in the New Superclass window and provide additional information in the Properties view; see Figure 8-58.



<Superclass Domain> HIGHLVL

Name: HIGHLVL
Comments:
[✓] Enable service class
Agent priority: -32768
Prefetch priority: DEFAULT
Work action set name: HIGHLVL
Operating system service class:
Work identities:

Figure 8-58 Create by Yourself - New Service Class properties view

After creating 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.

8.3.3 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. 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 a database and creates a new WLM scheme. Table 8-2 shows the DB2 objects that map to the Design Studio WLM entities.

Table 8-2 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 8-3.

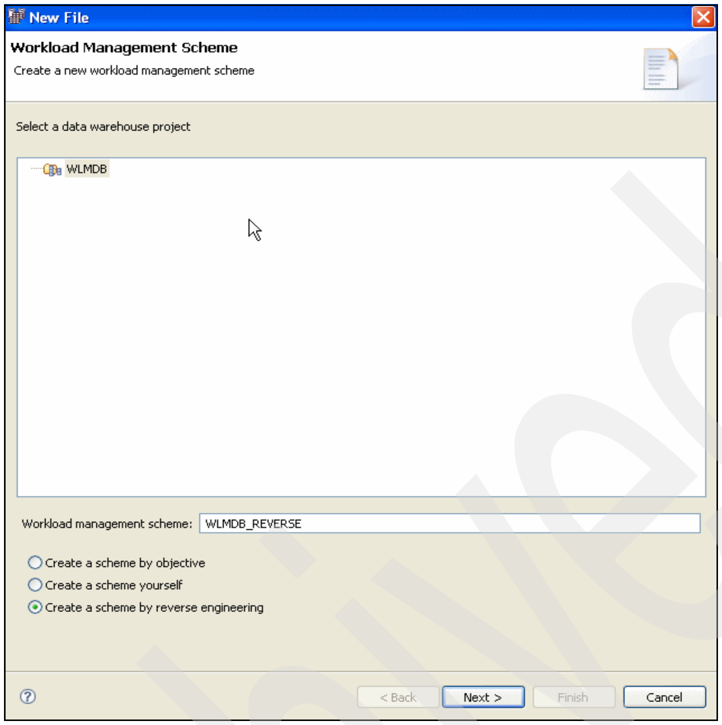
Table 8-3 Control rules mapping

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 8-59), type a unique name in the Workload management scheme field, select **Create a scheme by reverse engineering**, and then click **Next**.

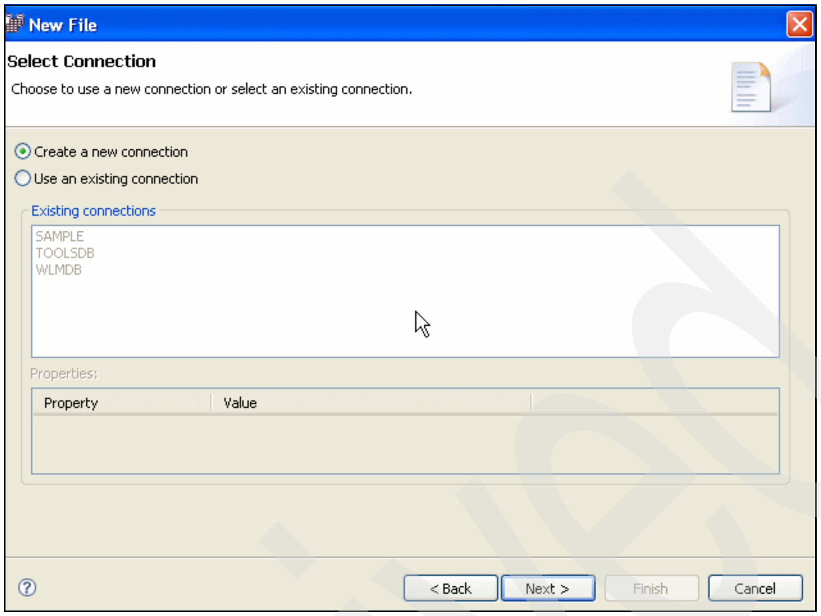

New File
Workload Management Scheme
Create a new workload management scheme
Select a data warehouse project
WLMDB
Workload management scheme:
< Back
Next >
Finish
Cancel

Workload management scheme: WLMDB_REVERSE

- Create a scheme by objective
- Create a scheme yourself
- [✓ Create a scheme by reverse engineering]

Figure 8-59 Workload Management Scheme - reverse engineering a scheme

3. In the Select Connection panel (Figure 8-60 on page 246), connect to the database where the collection of information about the scheme exists by using the existing connection, or create a new connection. We selected **Create a new connection**.



Select Connection

Choose to use a new connection or select an existing connection.

(●) Create a new connection
( ) Use an existing connection

Existing connections

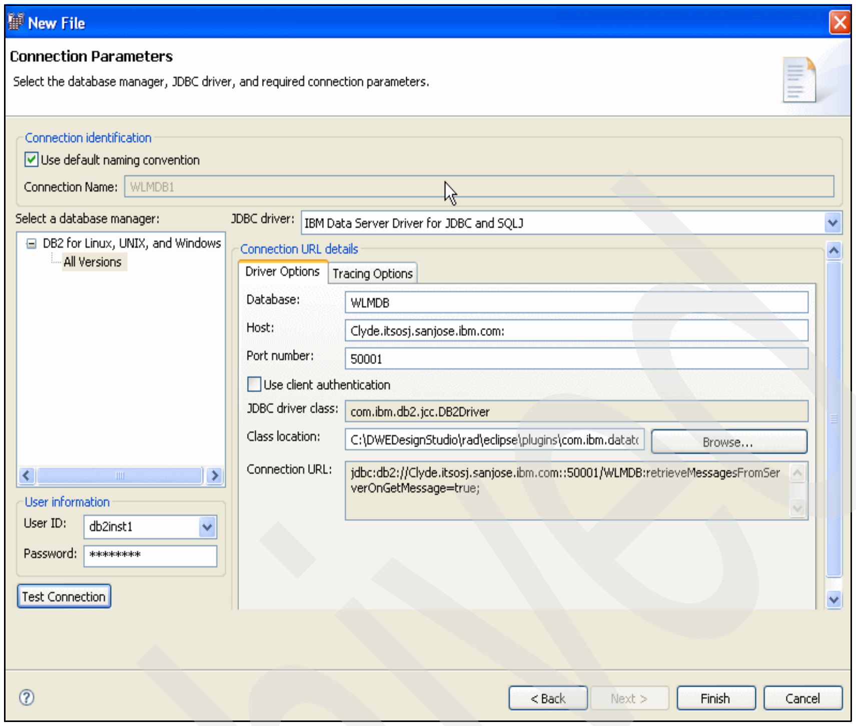
SAMPLE
TOOLSDB
WLMDB

Properties:

Property	Value

Figure 8-60 Create a new database connection

4. Complete the Connections Parameters (Figure 8-61 on page 247) panel, and click **Finish**.


New File
Connection Parameters
Select the database manager, JDBC driver, and required connection parameters.
Connection identification
Test Connection
< Back
Next >
Finish
Cancel

[✓] Use default naming convention

Connection Name: WLMDB1

Select a database manager: DB2 for Linux, UNIX, and Windows — All Versions

JDBC driver: IBM Data Server Driver for JDBC and SQLJ

Connection URL details — Driver Options | Tracing Options

Database:	WLMDB
Host:	Clyde.itsosj.sanjose.ibm.com:
Port number:	50001

[ ] Use client authentication

JDBC driver class:	com.ibm.db2.jcc.DB2Driver
Class location:	C:\DWEDesignStudio\rad\eclipse\plugins\com.ibm.datat
Connection URL:	jdbc:db2://Clyde.itsosj.sanjose.ibm.com::50001/WLMDB:retrieveMessagesFromServerOnGetMessage=true;

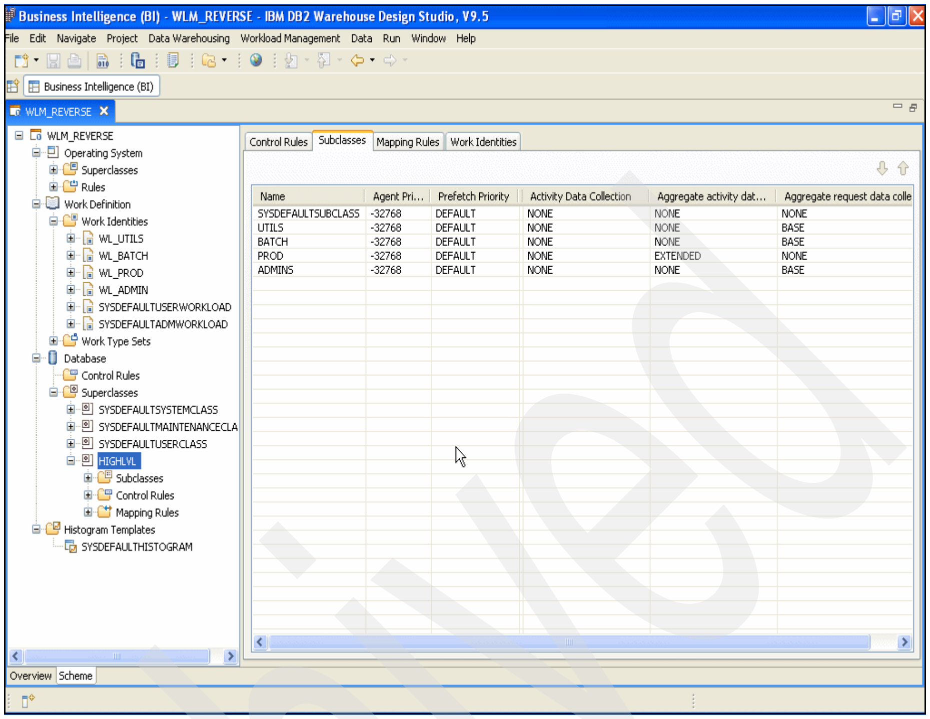
User information

User ID:	db2inst1
Password:	********

Figure 8-61 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 name is <WLM scheme name>.wlms. The output is shown in Figure 8-62 on page 248.

   Now, you can modify the scheme according to your specifications.



Name	Agent Pri...	Prefetch Priority	Activity Data Collection	Aggregate activity dat...	Aggregate request data colle
SYSDEFAULTSUBCLASS	-32768	DEFAULT	NONE	NONE	NONE
UTILS	-32768	DEFAULT	NONE	NONE	BASE
BATCH	-32768	DEFAULT	NONE	NONE	BASE
PROD	-32768	DEFAULT	NONE	EXTENDED	NONE
ADMINS	-32768	DEFAULT	NONE	NONE	BASE

Figure 8-62 WLM schema created from reverse engineering process

8.4 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, then select **Workload Management -> Validate**.
3. Select **Workload Management -> Generate Code**.

If you receive errors or warnings, open the Problems view to learn more about the problem description, then 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 the 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 display an 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

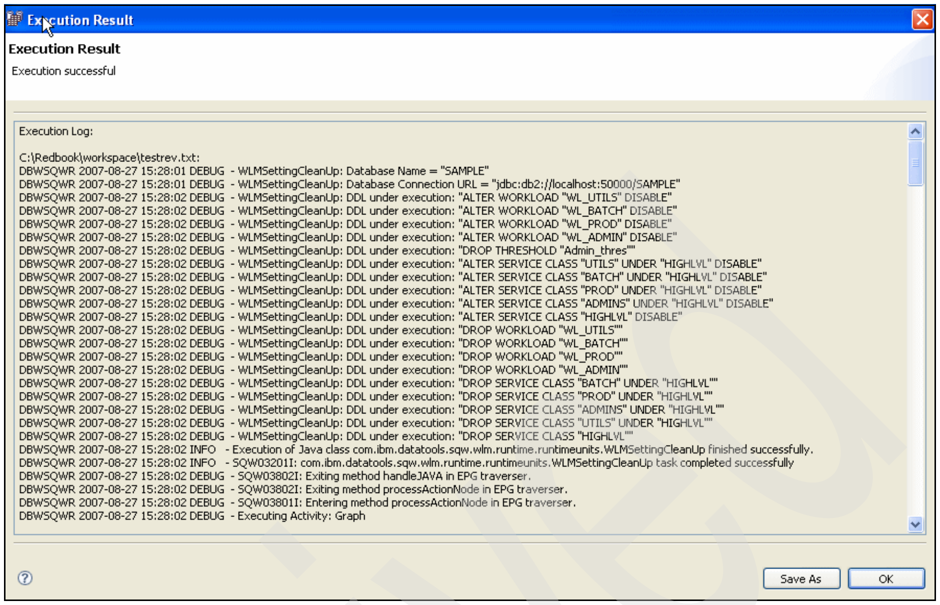
The execution approach is a *clean* execution. When you execute a workload management scheme, it begins by wiping off all the exiting WLM configuration on the target and creating the entire scheme fresh in the target. Note, however, that if the execution fails, then the settings for the WLM configuration that exist in the database are lost.

We recommend using this option only when you want to create 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**.
   - Or 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 8-63 shows the execution results panel using Execute.



Execution Result

Execution successful

```
Execution Log:

C:\Redbook\workspace\testrev.txt:
DBWSQWR 2007-08-27 15:28:01 DEBUG  - WLMSettingCleanUp: Database Name = "SAMPLE"
DBWSQWR 2007-08-27 15:28:01 DEBUG  - WLMSettingCleanUp: Database Connection URL = "jdbc:db2://localhost:50000/SAMPLE"
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "ALTER WORKLOAD "WL_UTILS" DISABLE"
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "ALTER WORKLOAD "WL_BATCH" DISABLE"
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "ALTER WORKLOAD "WL_PROD" DISABLE"
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "ALTER WORKLOAD "WL_ADMIN" DISABLE"
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "DROP THRESHOLD "Admin_thres""
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "ALTER SERVICE CLASS "UTILS" UNDER "HIGHLVL" DISABLE"
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "ALTER SERVICE CLASS "BATCH" UNDER "HIGHLVL" DISABLE"
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "ALTER SERVICE CLASS "PROD" UNDER "HIGHLVL" DISABLE"
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "ALTER SERVICE CLASS "ADMINS" UNDER "HIGHLVL" DISABLE"
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "ALTER SERVICE CLASS "HIGHLVL" DISABLE"
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "DROP WORKLOAD "WL_UTILS""
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "DROP WORKLOAD "WL_BATCH""
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "DROP WORKLOAD "WL_PROD""
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "DROP WORKLOAD "WL_ADMIN""
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "DROP SERVICE CLASS "BATCH" UNDER "HIGHLVL""
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "DROP SERVICE CLASS "PROD" UNDER "HIGHLVL""
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "DROP SERVICE CLASS "ADMINS" UNDER "HIGHLVL""
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "DROP SERVICE CLASS "UTILS" UNDER "HIGHLVL""
DBWSQWR 2007-08-27 15:28:02 DEBUG  - WLMSettingCleanUp: DDL under execution: "DROP SERVICE CLASS "HIGHLVL""
DBWSQWR 2007-08-27 15:28:02 INFO   - Execution of Java class com.ibm.datatools.sqw.wlm.runtime.runtimeunits.WLMSettingCleanUp finished successfully.
DBWSQWR 2007-08-27 15:28:02 INFO   - SQW03201I: com.ibm.datatools.sqw.wlm.runtime.runtimeunits.WLMSettingCleanUp task completed successfully
DBWSQWR 2007-08-27 15:28:02 DEBUG  - SQW03802I: Exiting method handleJAVA in EPG traverser.
DBWSQWR 2007-08-27 15:28:02 DEBUG  - SQW03802I: Exiting method processActionNode in EPG traverser.
DBWSQWR 2007-08-27 15:28:02 DEBUG  - SQW03801I: Entering method processActionNode in EPG traverser.
DBWSQWR 2007-08-27 15:28:02 DEBUG  - Executing Activity: Graph
```

Figure 8-63 Workload Management Execute - Execution Result screen

Delta Execution

Delta Execution performs ALTER whenever possible, and performs CREATE or DROP of WLM entities only when necessary. It performs reverse engineering on the target database and determines 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 statement 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 an error occurs during Delta Execution, Design Studio will not make any changes in the database. Your database will be in the same state as before the Delta Execution.

- Many dependencies between WLM entities.

  On a complex system with many WLM entities, the 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.

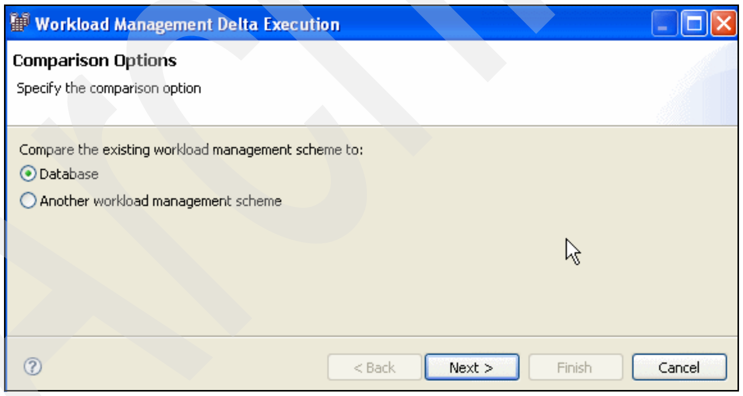
  Delta Execution avoids DROP/CREATE of WLM entities where possible. When executing a WLM scheme on a database, Design Studio performs ALTER to match the settings in the workload management scheme, and only performs DROP/CREATE if necessary.

We recommend Delta Execution for executing WLM entities because 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 panel, select **Workload Management -> Delta Execute.**
2. In the Comparison Options window (Figure 8-64), select **Database**, then click **Next**.



Workload Management Delta Execution

Comparison Options

Specify the comparison option

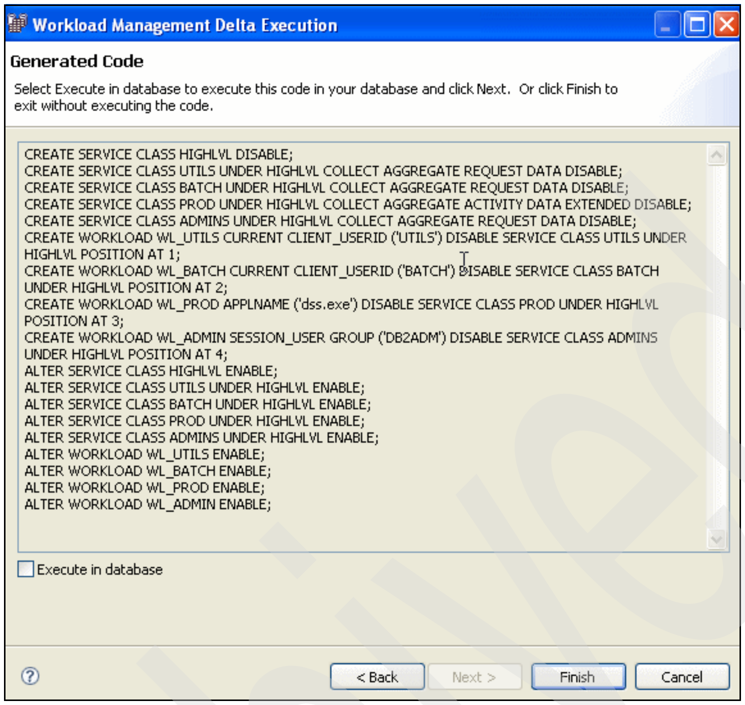
Compare the existing workload management scheme to:

[✓ Database]

Another workload management scheme

Figure 8-64 Delta Execution - Compare options

3. In the Select Connection window, select **Create a new connection** or **Use an existing connection**.
4. Delta execution compares the current WLM scheme with WLM entities in the database and generates a set of SQL statements; see Figure 8-65.



Generated Code

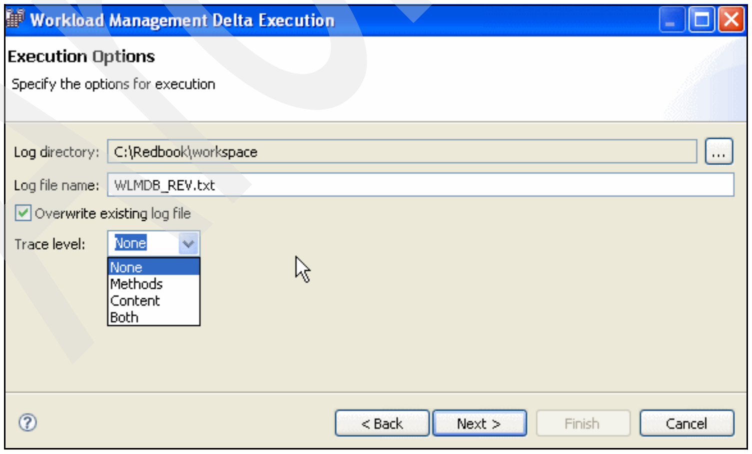
Select Execute in database to execute this code in your database and click Next. Or click Finish to exit without executing the code.

```
CREATE SERVICE CLASS HIGHLVL DISABLE;
CREATE SERVICE CLASS UTILS UNDER HIGHLVL COLLECT AGGREGATE REQUEST DATA DISABLE;
CREATE SERVICE CLASS BATCH UNDER HIGHLVL COLLECT AGGREGATE REQUEST DATA DISABLE;
CREATE SERVICE CLASS PROD UNDER HIGHLVL COLLECT AGGREGATE ACTIVITY DATA EXTENDED DISABLE;
CREATE SERVICE CLASS ADMINS UNDER HIGHLVL COLLECT AGGREGATE REQUEST DATA DISABLE;
CREATE WORKLOAD WL_UTILS CURRENT CLIENT_USERID ('UTILS') DISABLE SERVICE CLASS UTILS UNDER
HIGHLVL POSITION AT 1;
CREATE WORKLOAD WL_BATCH CURRENT CLIENT_USERID ('BATCH') DISABLE SERVICE CLASS BATCH
UNDER HIGHLVL POSITION AT 2;
CREATE WORKLOAD WL_PROD APPLNAME ('dss.exe') DISABLE SERVICE CLASS PROD UNDER HIGHLVL
POSITION AT 3;
CREATE WORKLOAD WL_ADMIN SESSION_USER GROUP ('DB2ADM') DISABLE SERVICE CLASS ADMINS
UNDER HIGHLVL POSITION AT 4;
ALTER SERVICE CLASS HIGHLVL ENABLE;
ALTER SERVICE CLASS UTILS UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS BATCH UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS PROD UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS ADMINS UNDER HIGHLVL ENABLE;
ALTER WORKLOAD WL_UTILS ENABLE;
ALTER WORKLOAD WL_BATCH ENABLE;
ALTER WORKLOAD WL_PROD ENABLE;
ALTER WORKLOAD WL_ADMIN ENABLE;
```

☐ Execute in database

Figure 8-65 Delta execution - Generated Code

5. In the Generated Code window, select **Execute in database** and click **Next**.
6. In the Execution Options window (Figure 8-66 on page 252), specify any options and click **Next**.



Execution Options

Specify the options for execution

Log directory: C:\Redbook\workspace

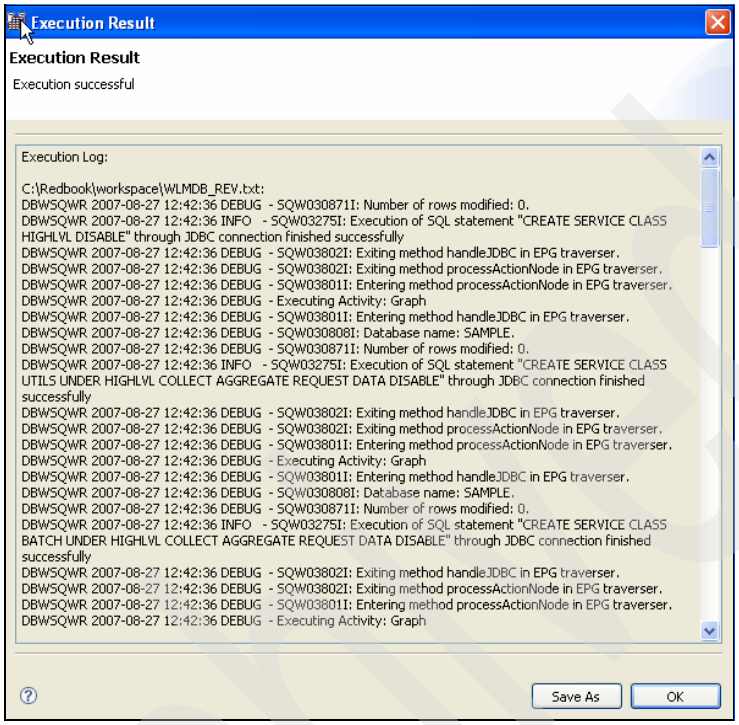
Log file name: WLMDB_REV.txt

☑ Overwrite existing log file

Trace level: None

Figure 8-66 Delta Execution - Execute options screen

- In the Execution Result window (Figure 8-67), verify the result and click **Finish**.



```
Execution Log:

C:\Redbook\workspace\WLMDB_REV.txt:
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW030871I: Number of rows modified: 0.
DBWSQWR 2007-08-27 12:42:36 INFO   - SQW03275I: Execution of SQL statement "CREATE SERVICE CLASS
HIGHLVL DISABLE" through JDBC connection finished successfully
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW03802I: Exiting method handleJDBC in EPG traverser.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW03802I: Exiting method processActionNode in EPG traverser.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW03801I: Entering method processActionNode in EPG traverser.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - Executing Activity: Graph
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW03801I: Entering method handleJDBC in EPG traverser.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW030808I: Database name: SAMPLE.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW030871I: Number of rows modified: 0.
DBWSQWR 2007-08-27 12:42:36 INFO   - SQW03275I: Execution of SQL statement "CREATE SERVICE CLASS
UTILS UNDER HIGHLVL COLLECT AGGREGATE REQUEST DATA DISABLE" through JDBC connection finished
successfully
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW03802I: Exiting method handleJDBC in EPG traverser.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW03802I: Exiting method processActionNode in EPG traverser.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW03801I: Entering method processActionNode in EPG traverser.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - Executing Activity: Graph
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW03801I: Entering method handleJDBC in EPG traverser.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW030808I: Database name: SAMPLE.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW030871I: Number of rows modified: 0.
DBWSQWR 2007-08-27 12:42:36 INFO   - SQW03275I: Execution of SQL statement "CREATE SERVICE CLASS
BATCH UNDER HIGHLVL COLLECT AGGREGATE REQUEST DATA DISABLE" through JDBC connection finished
successfully
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW03802I: Exiting method handleJDBC in EPG traverser.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW03802I: Exiting method processActionNode in EPG traverser.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - SQW03801I: Entering method processActionNode in EPG traverser.
DBWSQWR 2007-08-27 12:42:36 DEBUG  - Executing Activity: Graph
```

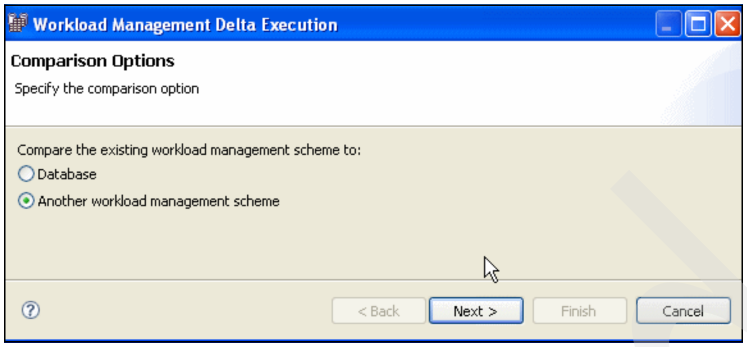
Figure 8-67 Delta Execution - Results screen

Delta Execute against WLM schemes

Use Delta Execute to compare two WLM schemes. This can be useful in determining what WLM changes have been made on the database if different WLM scheme versions are available.

To use the Delta Execute option, follow these steps:

1. In the workload management scheme open, select **Workload Management -> Delta Execute.**
2. In the Comparison Options window, select **Another workload management scheme**, then click **Next**; see Figure 8-68.


Workload Management Delta Execution

Comparison Options

Specify the comparison option

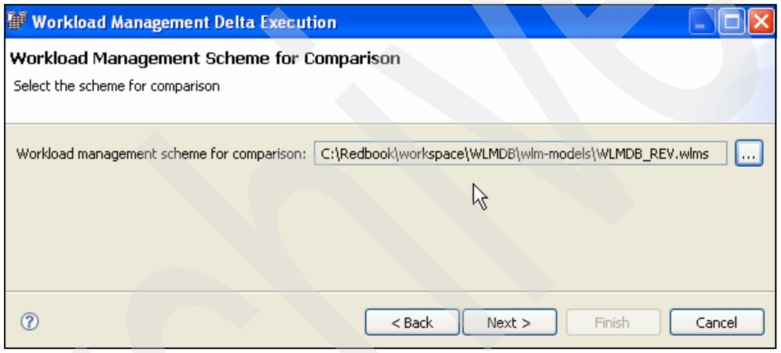
Compare the existing workload management scheme to:

( ) Database

[✓ Another workload management scheme]

Figure 8-68 Delta Execution - Another workload management scheme

3. Choose the .wlm files of the WLM scheme to be compared with; see Figure 8-69.


Workload Management Delta Execution

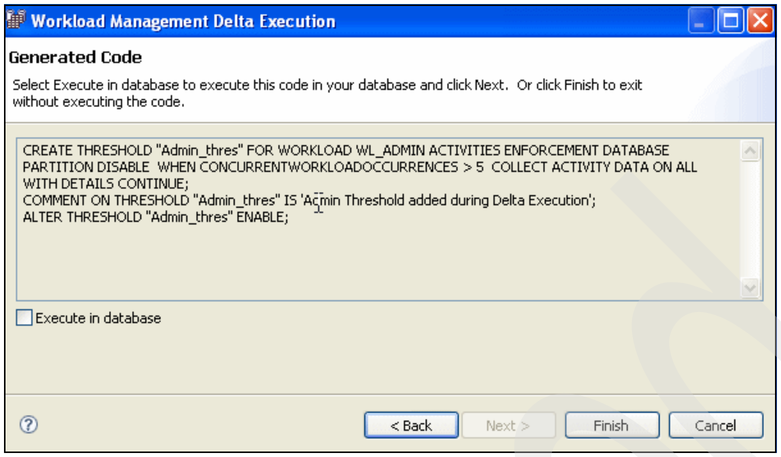
Workload Management Scheme for Comparison

Select the scheme for comparison

Workload management scheme for comparison: C:\Redbook\workspace\WLMDB\wlm-models\WLMDB_REV.wlms

Figure 8-69 Delta Execute - Select WLM scheme for comparison

4. The delta SQL statement is shown in the Generated Code window (Figure 8-70).


Workload Management Delta Execution
Generated Code
Select Execute in database to execute this code in your database and click Next. Or click Finish to exit without executing the code.
Execute in database
< Back
Next >
Finish
Cancel

```
CREATE THRESHOLD "Admin_thres" FOR WORKLOAD WL_ADMIN ACTIVITIES ENFORCEMENT DATABASE
PARTITION DISABLE  WHEN CONCURRENTWORKLOADOCCURRENCES > 5  COLLECT ACTIVITY DATA ON ALL
WITH DETAILS CONTINUE;
COMMENT ON THRESHOLD "Admin_thres" IS 'Admin Threshold added during Delta Execution';
ALTER THRESHOLD "Admin_thres" ENABLE;
```

Figure 8-70 Delta Execute - Compare WLM Schemes results

8.5 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. As previously noted, in DB2 9.5, you can utilize the AIX WLM to manage only CPU utilization.

8.5.1 Creating operating system service classes and limits

Using Design Studio, you can create operating system service classes when a new WLM scheme is created, or by modifying the existing WLM scheme using the tree view.

Creating operating system service classes on an existing DB2 WLM scheme

Before you begin using Design Studio to create AIX WLM service classes, it is assumed that you have the following created:

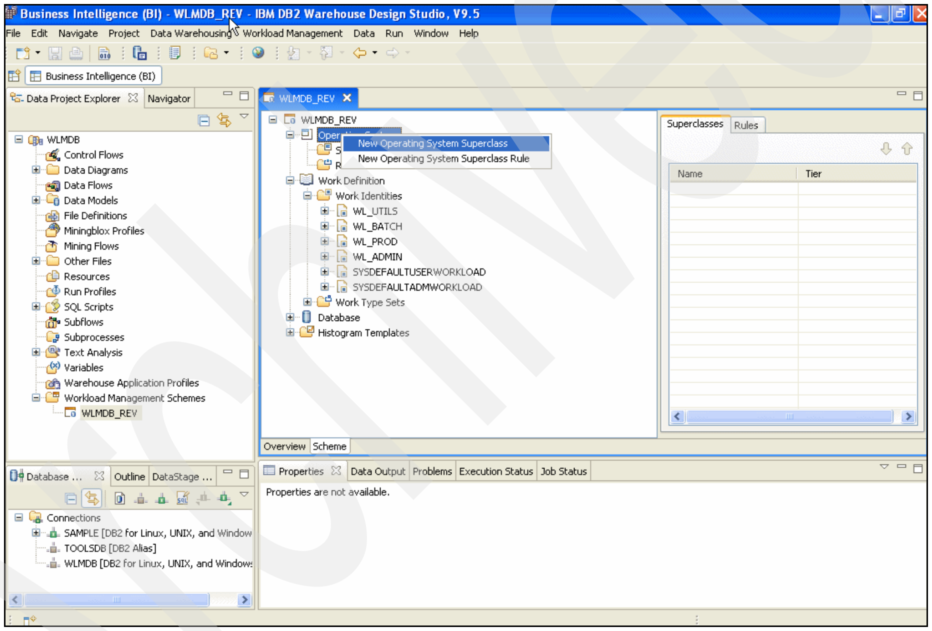
- A WLM scheme
- DB2 work identities, superclasses, and subclasses

Plan the mapping between DB2 and AIX WLM mapping scheme (for example, 1:1 or flat mapping). We recommend that you start with 1:1 mapping and then expand.

Create operating system superclasses

Follow these steps to create AIX WLM superclasses:

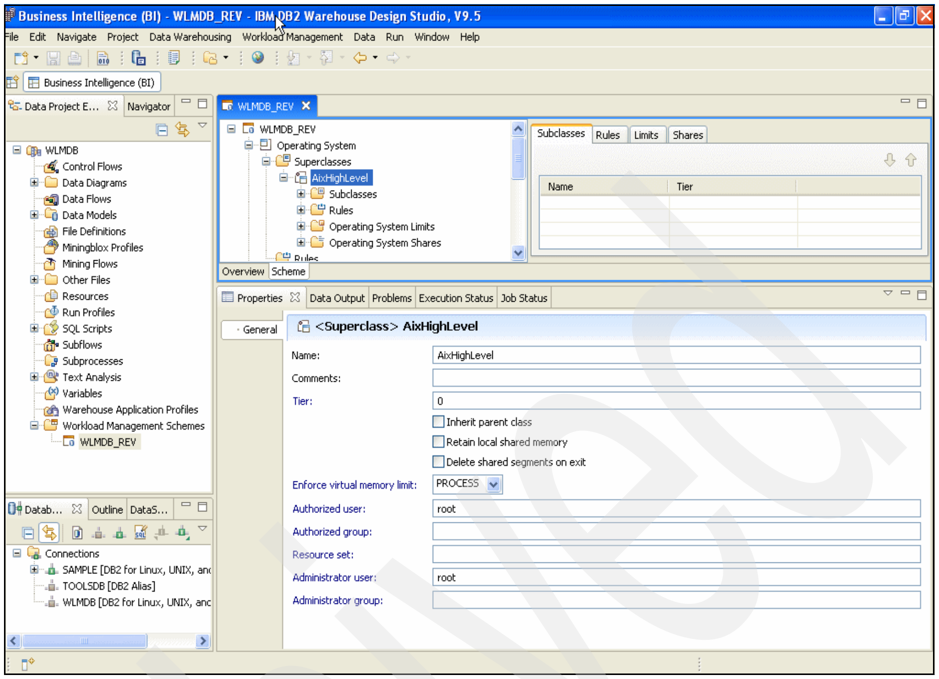
1. In the Data Project Explorer, expand the data warehouse project that contains the scheme that you want to work with.
2. Expand the Workload Management Schemes folder.
3. Double-click the scheme for which you are creating an operating system service class, then expand the scheme.
4. Right-click **Operating System**, then select **New Operating System Superclass**.
5. In the New Operating System Service Class window (Figure 8-71), type a unique name and click **OK**.



Properties are not available.

Figure 8-71 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 8-72.



<Superclass> AixHighLevel

Name:	AixHighLevel
Comments:	
Tier:	0
	☐ Inherit parent class
	☐ Retain local shared memory
	☐ Delete shared segments on exit
Enforce virtual memory limit:	PROCESS
Authorized user:	root
Authorized group:	
Resource set:	
Administrator user:	root
Administrator group:	

Business Intelligence (BI) - WLMDB_REV - IBM DB2 Warehouse Design Studio, V9.5
File Edit Navigate Project Data Warehousing Workload Management Data Run Window Help
Business Intelligence (BI)
Data Project E... Navigator
WLMDB
Control Flows
Data Diagrams
Data Flows
Data Models
File Definitions
Miningblox Profiles
Mining Flows
Other Files
Resources
Run Profiles
SQL Scripts
Subflows
Subprocesses
Text Analysis
Variables
Warehouse Application Profiles
Workload Management Schemes
WLMDB_REV
Datab... Outline DataS...
Connections
SAMPLE [DB2 for Linux, UNIX, and
TOOLSDB [DB2 Alias]
WLMDB [DB2 for Linux, UNIX, and
WLMDB_REV
WLMDB_REV
Operating System
Superclasses
AixHighLevel
Subclasses
Rules
Operating System Limits
Operating System Shares
Rules
Overview Scheme
Subclasses Rules Limits Shares
Name Tier
Properties Data Output Problems Execution Status Job Status
General

Figure 8-72 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.

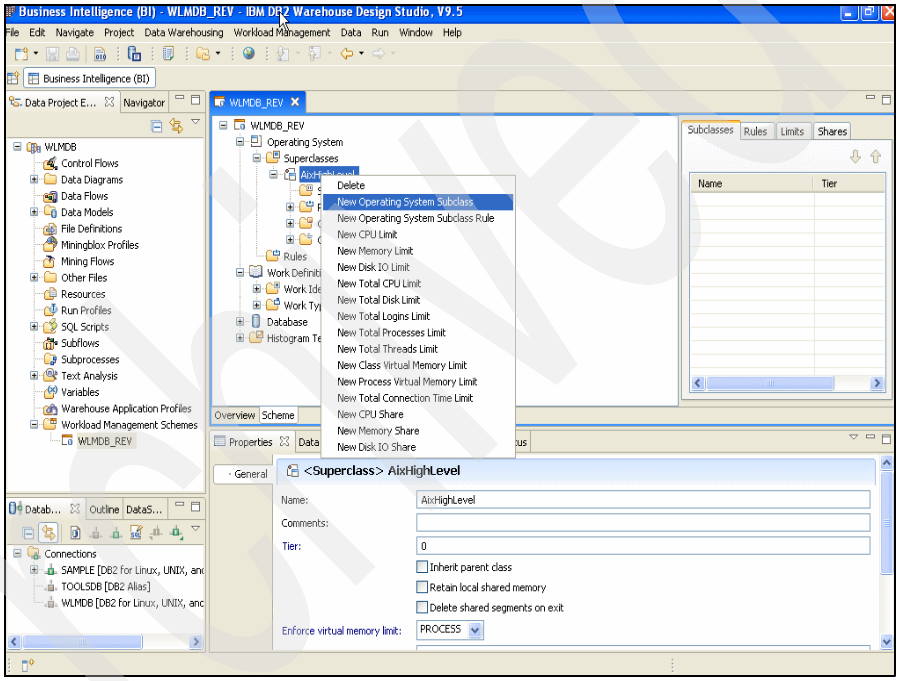
Create operating system subclass

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

Follow these steps to create an operating system subclass:

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 then expand the scheme in the Scheme view.
3. Expand the operating system and then expand the Superclasses folder.
4. Right-click the appropriate superclass and select **New Operating System Subclass**; see Figure 8-73.



<Superclass> AixHighLevel

Name: AixHighLevel

Comments:

Tier: 0

[ ] Inherit parent class

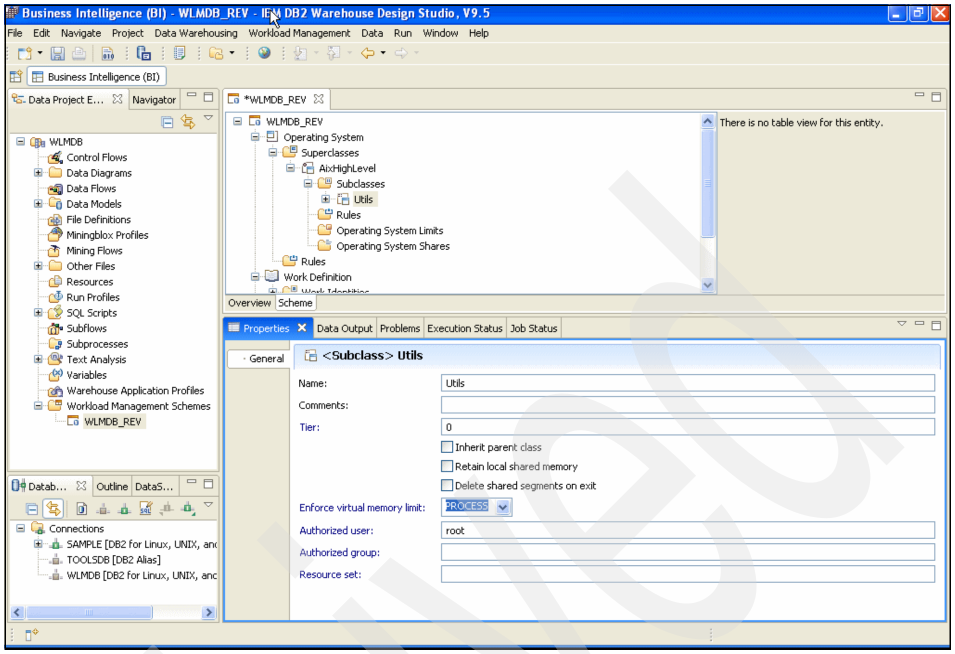
[ ] Retain local shared memory

[ ] Delete shared segments on exit

Enforce virtual memory limit: PROCESS

Figure 8-73 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 8-74.


Business Intelligence (BI) - WLMDB_REV - IBM DB2 Warehouse Design Studio, V9.5
File Edit Navigate Project Data Warehousing Workload Management Data Run Window Help
Business Intelligence (BI)
Data Project E... Navigator
WLMDB
Control Flows
Data Diagrams
Data Flows
Data Models
File Definitions
Miningblox Profiles
Mining Flows
Other Files
Resources
Run Profiles
SQL Scripts
Subflows
Subprocesses
Text Analysis
Variables
Warehouse Application Profiles
Workload Management Schemes
WLMDB_REV
*WLMDB_REV
WLMDB_REV
Operating System
Superclasses
AixHighLevel
Subclasses
Utils
Rules
Operating System Limits
Operating System Shares
Rules
Work Definition
There is no table view for this entity.
Overview Scheme
Datab... Outline DataS...
Connections
SAMPLE [DB2 for Linux, UNIX, and
TOOLSDB [DB2 Alias]
WLMDB [DB2 for Linux, UNIX, and
Properties Data Output Problems Execution Status Job Status
General

<Subclass> Utils

Name: Utils

Comments:

Tier: 0

☐ Inherit parent class

☐ Retain local shared memory

☐ Delete shared segments on exit

Enforce virtual memory limit: PROCESS

Authorized user: root

Authorized group:

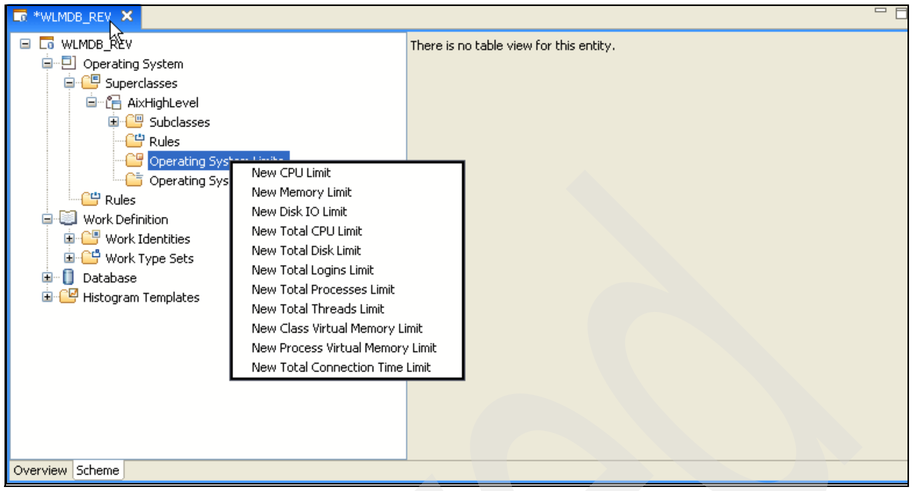
Resource set:

Figure 8-74 Operating system subclass - Properties view

Limiting system resources for operating system service classes

Follow these steps to create a limit for a system resource:

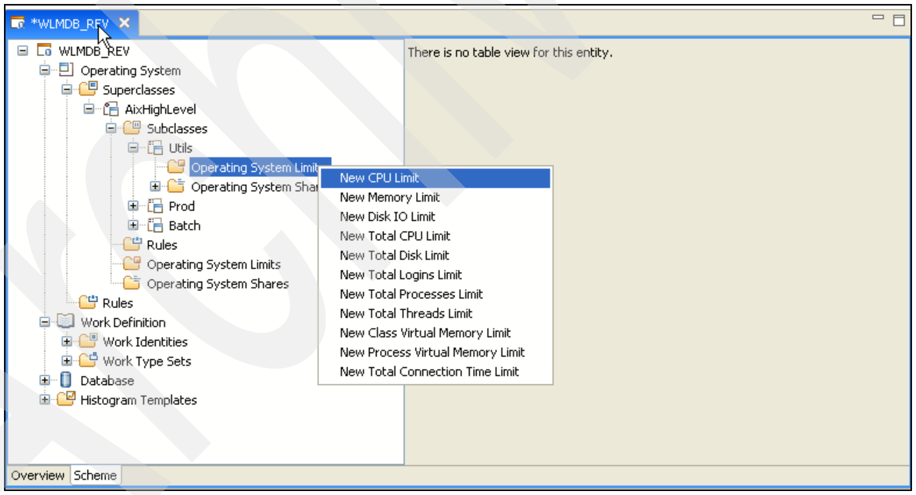
1. In the Data Project Explorer, expand the data warehouse project that contains the scheme that you want to work with, and then expand the Workload Management Schemes folder.
2. Double-click the scheme for which you are creating an assignment rule.
3. In the Scheme view, expand the scheme, Operating System, Superclasses, and the appropriate superclass. Right-click **Operating System Limits** at the superclass level, then select the system resource; see Figure 8-75 on page 260.



There is no table view for this entity.

Figure 8-75 Operating system limits - superclass

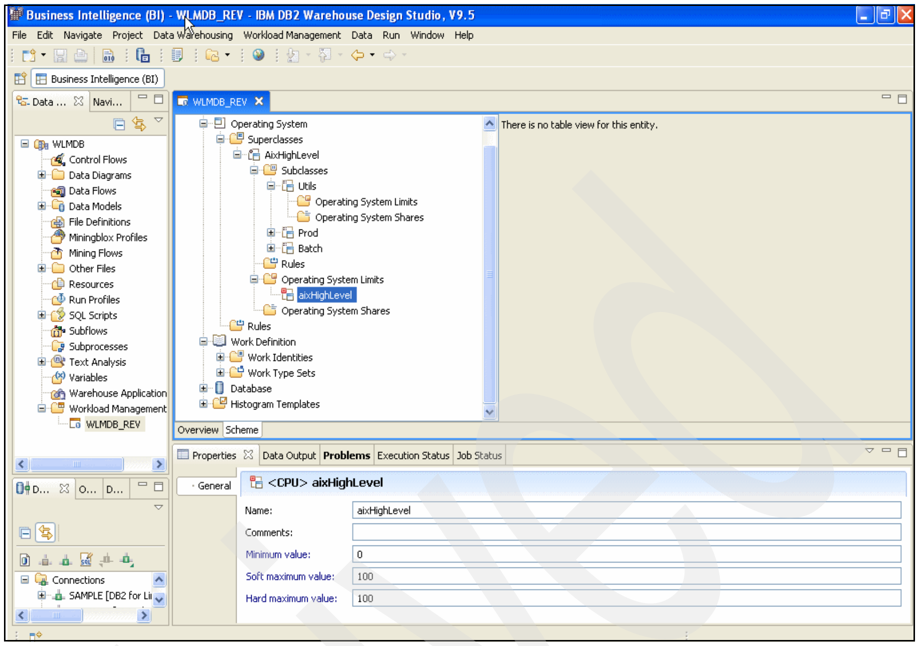
You also can create the operating system limits on the subclass level by selecting the resource on the subclass level, as shown in Figure 8-76.



There is no table view for this entity.

Figure 8-76 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 8-77 on page 261.



There is no table view for this entity.

<CPU> aixHighLevel

Name: aixHighLevel
Comments:
Minimum value: 0
Soft maximum value: 100
Hard maximum value: 100

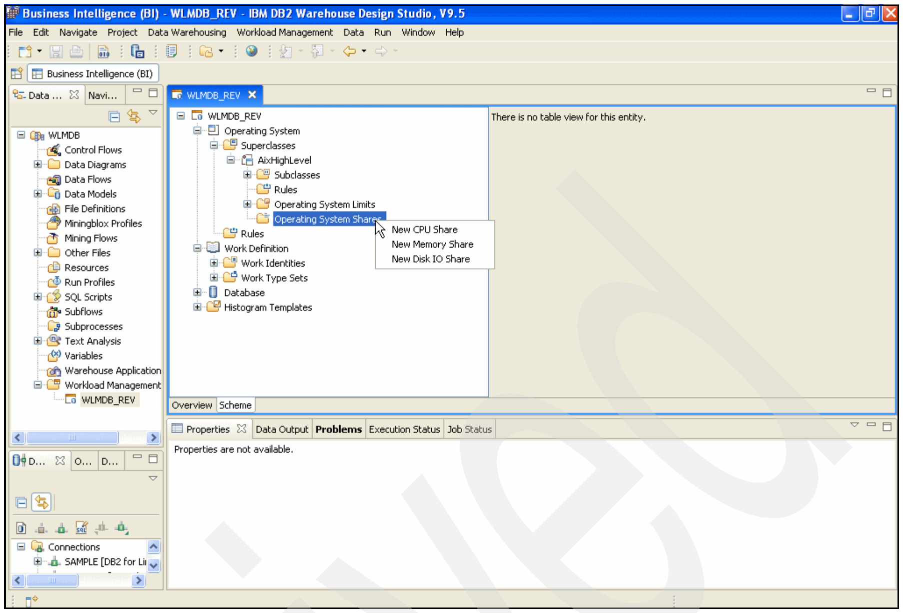
Figure 8-77 Create Operating System limits - Properties view

Allocating resource shares to operating system service classes

Follow these steps to create a share of a system resource:

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 8-78 on page 262.

   You also can create operating system share on the subclass level by selecting Operating System Shares at the subclass level.

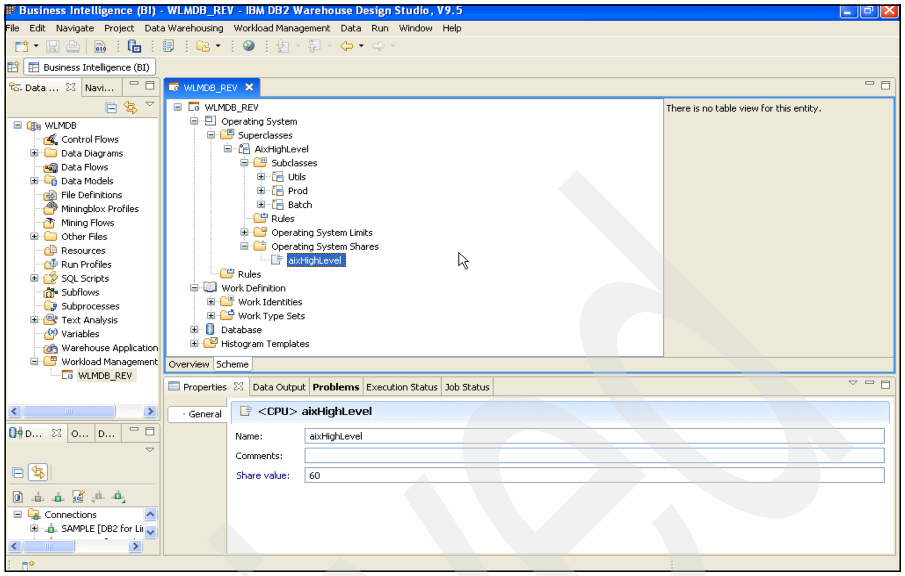


There is no table view for this entity.

Properties are not available.

Figure 8-78   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 8-79 on page 263.



There is no table view for this entity.

<CPU> aixHighLevel

Name: aixHighLevel

Comments:

Share value: 60

Figure 8-79   Create Operating System share - Superclass Properties view

8.5.2 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 a DB2 superclass with an AIX superclass

Follow these steps to associate a DB2 superclass with an AIX superclass:

1. In the Scheme view, expand the scheme, Database, and Superclasses, and then select the superclass you want to associate with the operating system superclass.

- In the General tab of the Properties view for the DB2 superclass, associate with the AIX service class by selecting the ... button for Operating system service class. Figure 8-80 on page 264 shows an example of associating a DB2 superclass with an AIX superclass.

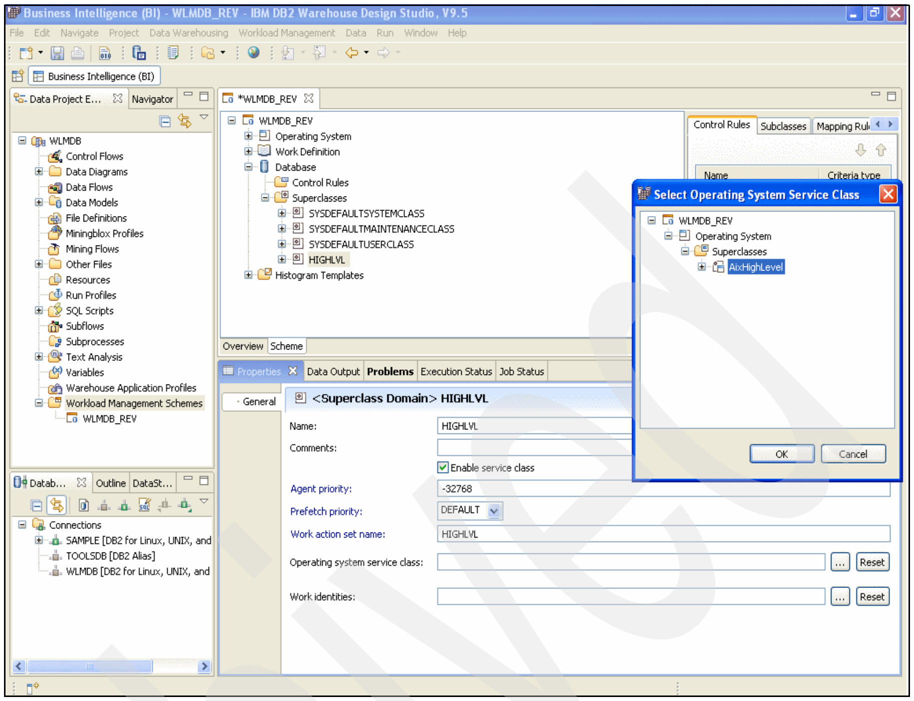

Business Intelligence (BI) - WLMDB_REV - IBM DB2 Warehouse Design Studio, V9.5
File Edit Navigate Project Data Warehousing Workload Management Data Run Window Help
Business Intelligence (BI)
Data Project E... Navigator
WLMDB
Control Flows
Data Diagrams
Data Flows
Data Models
File Definitions
Miningblox Profiles
Mining Flows
Other Files
Resources
Run Profiles
SQL Scripts
Subflows
Subprocesses
Text Analysis
Variables
Warehouse Application Profiles
Workload Management Schemes
WLMDB_REV
Datab... Outline DataSt...
Connections
SAMPLE [DB2 for Linux, UNIX, and
TOOLSDB [DB2 Alias]
WLMDB [DB2 for Linux, UNIX, and
*WLMDB_REV
WLMDB_REV
Operating System
Work Definition
Database
Control Rules
Superclasses
SYSDEFAULTSYSTEMCLASS
SYSDEFAULTMAINTENANCECLASS
SYSDEFAULTUSERCLASS
HIGHLVL
Histogram Templates
Overview Scheme
Control Rules Subclasses Mapping Rul
Name Criteria type
Select Operating System Service Class
WLMDB_REV
Operating System
Superclasses
AixHighLevel
OK Cancel
Properties Data Output Problems Execution Status Job Status
General
<Superclass Domain> HIGHLVL
... Reset
... Reset
Name: HIGHLVL
Comments:
[✓] Enable service class
Agent priority: -32768
Prefetch priority: DEFAULT
Work action set name: HIGHLVL
Operating system service class:
Work identities:

Figure 8-80 AIX superclass and DB2 superclass association

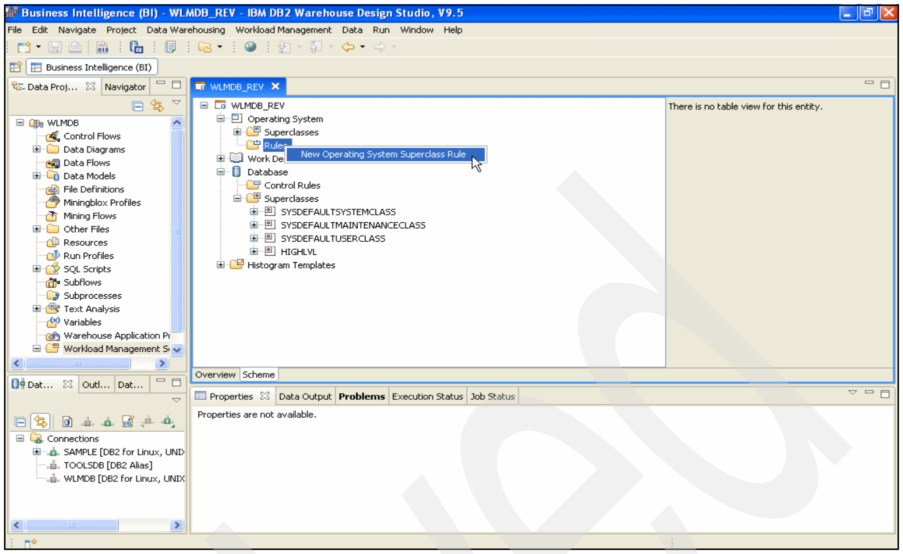
Adding control rules to AIX service classes

After the service classes association is completed, you can define the control rules on AIX service classes to manage the resource.

Follow these steps to add the control rules:

1. In the Scheme view, expand the scheme and Operating System, right-click **Rules** at the superclass level, then select **New Operating System Superclass Rule**; see Figure 8-81 on page 265.

   To add rules for a subclass, expand the view to the subclass level and right-lick **Rules** at the subclass level.



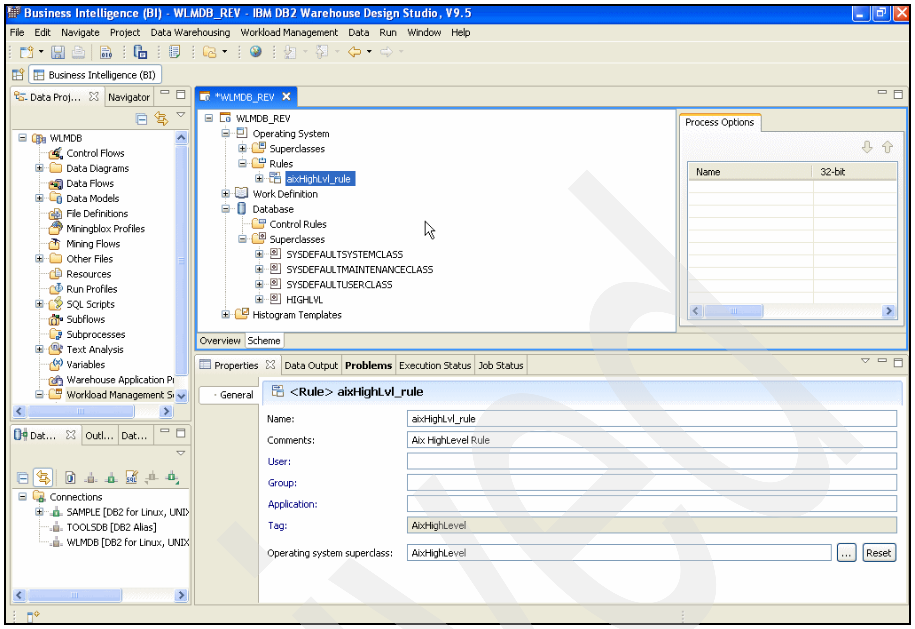
There is no table view for this entity.

Properties are not available.

Figure 8-81 Create New Operating System Superclass Rule screen

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 the operating system superclass, select the ... button for the operating system service class and select the superclass you want.

Figure 8-82 shows that, in our case, we created aixHighLvl_rule and associated it with operating system superclass AIXHighLevel.



Name: aixHighLvl_rule
Comments: Aix HighLevel Rule
User:
Group:
Application:
Tag: AixHighLevel
Operating system superclass: AixHighLevel

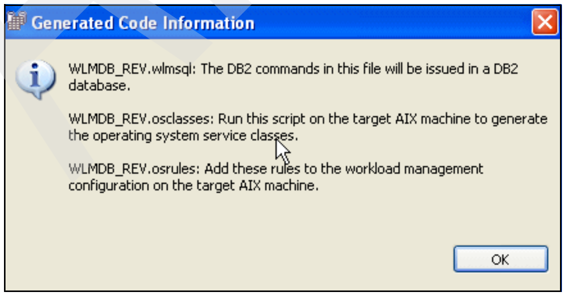
Figure 8-82 Operating system rule for superclass

Note: In Design Studio, Application Tag is a read-only field. Design Studio generates a string value for the 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 8-83 shows the confirmation screen from Generate code execution.



WLMDB_REV.wlmsql: The DB2 commands in this file will be issued in a DB2 database.

WLMDB_REV.osclasses: Run this script on the target AIX machine to generate the operating system service classes.

WLMDB_REV.osrules: Add these rules to the workload management configuration on the target AIX machine.

Figure 8-83 Generated Code Information screen

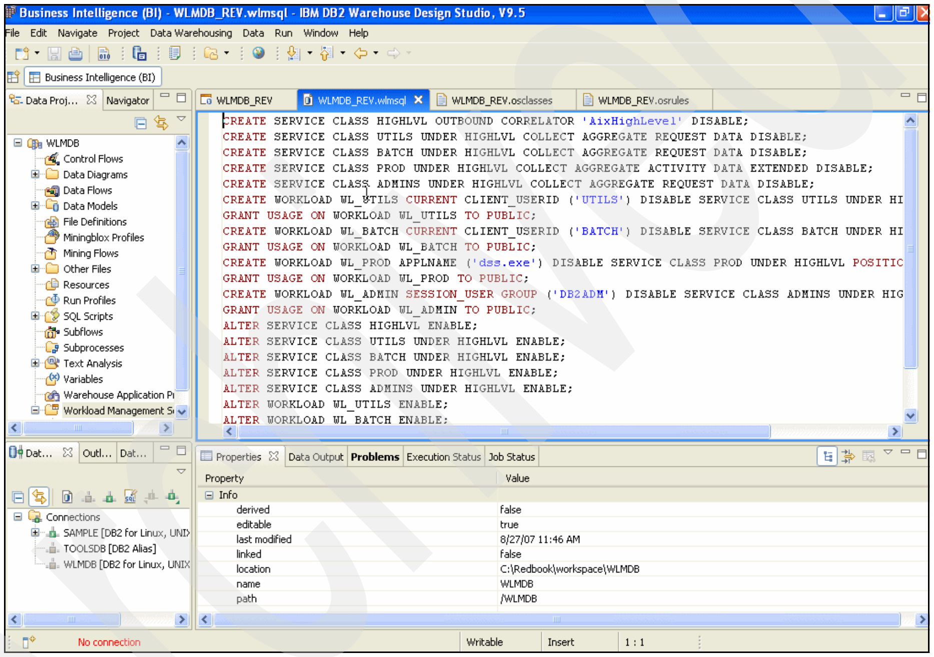
The Generated Code Information window lists the files that are created and their usage. If you click **OK**, Design Studio will automatically open the files generated in the Editors view.

Any of three possible files can be generated:

- 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 the Scheme name and a .wlmsql extension (for example, WLMDB_REV.wlmsql).

  Figure 8-84 shows the DB2 SQL generated by Design Studio.


Business Intelligence (BI) - WLMDB_REV.wlmsql - IBM DB2 Warehouse Design Studio, V9.5
File Edit Navigate Project Data Warehousing Data Run Window Help
Business Intelligence (BI)
Data Proj... Navigator
WLMDB_REV WLMDB_REV.wlmsql WLMDB_REV.osclasses WLMDB_REV.osrules
WLMDB
Control Flows
Data Diagrams
Data Flows
Data Models
File Definitions
Miningblox Profiles
Mining Flows
Other Files
Resources
Run Profiles
SQL Scripts
Subflows
Subprocesses
Text Analysis
Variables
Warehouse Application Pr
Workload Management Sc
Dat... Outl... Dat...
Connections
SAMPLE [DB2 for Linux, UNIX
TOOLSDB [DB2 Alias]
WLMDB [DB2 for Linux, UNIX
Properties Data Output Problems Execution Status Job Status
No connection
Writable Insert 1 : 1

```
CREATE SERVICE CLASS HIGHLVL OUTBOUND CORRELATOR 'AixHighLevel' DISABLE;
CREATE SERVICE CLASS UTILS UNDER HIGHLVL COLLECT AGGREGATE REQUEST DATA DISABLE;
CREATE SERVICE CLASS BATCH UNDER HIGHLVL COLLECT AGGREGATE REQUEST DATA DISABLE;
CREATE SERVICE CLASS PROD UNDER HIGHLVL COLLECT AGGREGATE ACTIVITY DATA EXTENDED DISABLE;
CREATE SERVICE CLASS ADMINS UNDER HIGHLVL COLLECT AGGREGATE REQUEST DATA DISABLE;
CREATE WORKLOAD WL_UTILS CURRENT CLIENT_USERID ('UTILS') DISABLE SERVICE CLASS UTILS UNDER HI
GRANT USAGE ON WORKLOAD WL_UTILS TO PUBLIC;
CREATE WORKLOAD WL_BATCH CURRENT CLIENT_USERID ('BATCH') DISABLE SERVICE CLASS BATCH UNDER HI
GRANT USAGE ON WORKLOAD WL_BATCH TO PUBLIC;
CREATE WORKLOAD WL_PROD APPLNAME ('dss.exe') DISABLE SERVICE CLASS PROD UNDER HIGHLVL POSITIC
GRANT USAGE ON WORKLOAD WL_PROD TO PUBLIC;
CREATE WORKLOAD WL_ADMIN SESSION_USER GROUP ('DB2ADM') DISABLE SERVICE CLASS ADMINS UNDER HIG
GRANT USAGE ON WORKLOAD WL_ADMIN TO PUBLIC;
ALTER SERVICE CLASS HIGHLVL ENABLE;
ALTER SERVICE CLASS UTILS UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS BATCH UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS PROD UNDER HIGHLVL ENABLE;
ALTER SERVICE CLASS ADMINS UNDER HIGHLVL ENABLE;
ALTER WORKLOAD WL_UTILS ENABLE;
ALTER WORKLOAD WL_BATCH ENABLE;
```

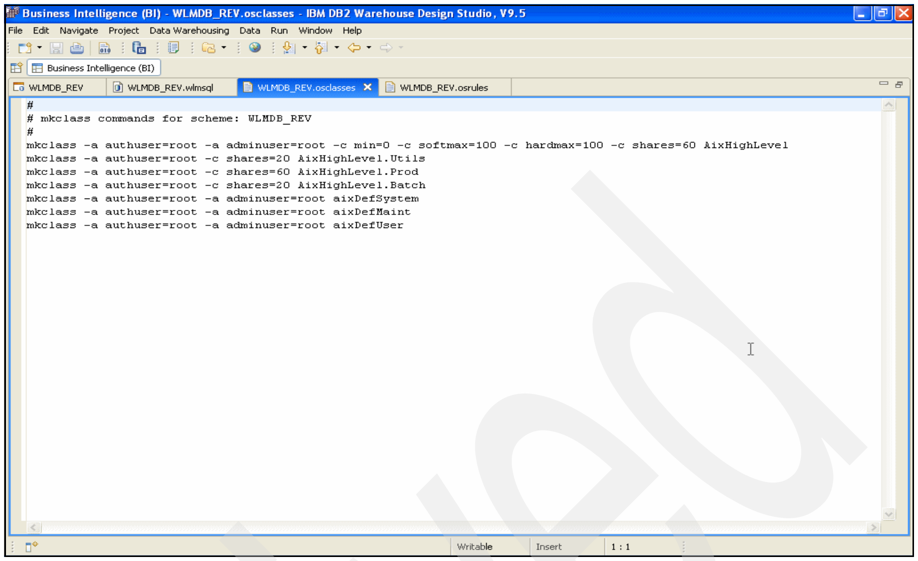
Property	Value
Info	
derived	false
editable	true
last modified	8/27/07 11:46 AM
linked	false
location	C:\Redbook\workspace\WLMDB
name	WLMDB
path	/WLMDB

Figure 8-84 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 the scheme name and .osclasses extension; for example, WLMDB_REV.osclasses.

  Figure 8-85 shows the sample output of the 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.wlmsql
WLMDB_REV.osclasses
WLMDB_REV.osrules
Writable
Insert
1 : 1

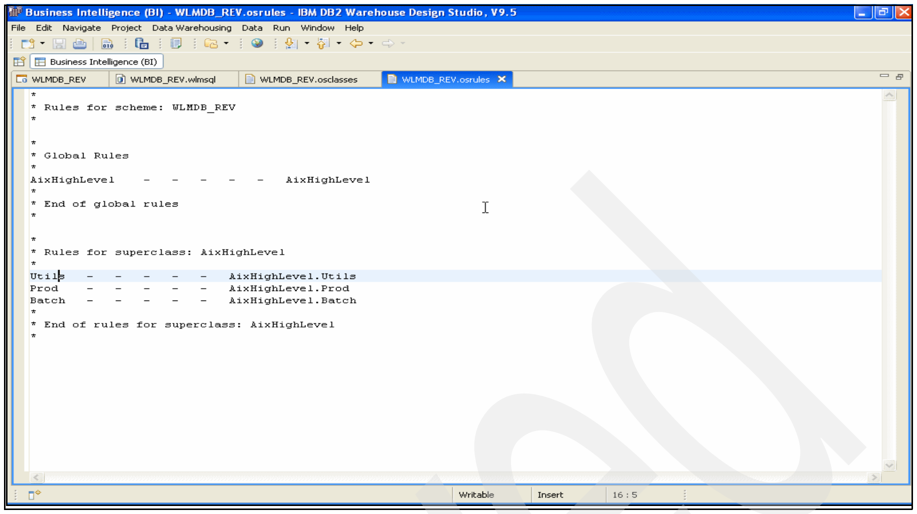
```
#
# mkclass commands for scheme: WLMDB_REV
#
mkclass -a authuser=root -a adminuser=root -c min=0 -c softmax=100 -c hardmax=100 -c shares=60 AixHighLevel
mkclass -a authuser=root -c shares=20 AixHighLevel.Utils
mkclass -a authuser=root -c shares=60 AixHighLevel.Prod
mkclass -a authuser=root -c shares=20 AixHighLevel.Batch
mkclass -a authuser=root -a adminuser=root aixDefSystem
mkclass -a authuser=root -a adminuser=root aixDefMaint
mkclass -a authuser=root -a adminuser=root aixDefUser
```

Figure 8-85 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 the scheme name and .osrules extension; for example, WLMDB_REV.osrules.

  Figure 8-86 on page 269 shows sample output of the operating systems rules file.



```
*
* Rules for scheme: WLMDB_REV
*


*
* Global Rules
*
AixHighLevel    -    -    -    -    -    AixHighLevel
*
* End of global rules
*


*
* Rules for superclass: AixHighLevel
*
Utils    -    -    -    -    -    AixHighLevel.Utils
Prod     -    -    -    -    -    AixHighLevel.Prod
Batch    -    -    -    -    -    AixHighLevel.Batch
*
* End of rules for superclass: AixHighLevel
*
```

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.wlmsql
WLMDB_REV.osclasses
WLMDB_REV.osrules
Writable
Insert
16 : 5

Figure 8-86 Generated code sample for operating system rules file

Note: DB2 WLM artifact code generated by Design Studio is executed by Design Studio against the target database. AIX WLM artifact 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 the appropriate connection and authority, or it can be shipped to the target source where you want to execute the WLM scheme.

9

DB2 Workload Manager and DB2 Performance Expert

This chapter explains how to use DB2 Performance Expert in coordination with DB2 Workload Manager functions.

As described in earlier chapters of this book, DB2 9.5 provides various ways to get information about your workload management definitions and statistics about applications running within the workload management 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, rather than 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.

We discuss the following topics in this chapter:

- An overview of DB2 Performance Expert (PE)
- Installing and configuring DB2 PE
- Monitoring your DB2 environment
- Monitoring DB2 WLM
- DB2 PE technical information

9.1 DB2 Performance Expert overview

When thinking 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.

Focus

This chapter discusses only the use of DB2 Performance Expert to monitor DB2 Workload Manager capabilities. To learn more about general usage of DB2 Performance Expert, consult the following sources:

- 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.fpeic.doc.ug/peic_home.htm

- IBM Redbooks publication *DB2 Performance Expert for Multiplatforms V2.2*, SG24-6470

How to get DB2 Performance Expert

DB2 Performance Expert is 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, Query Patroller, and Workload Manager.

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 refer to the DB2 Performance Expert home page:

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 detailed installation and configuration information, refer to *DB2 Performance Expert for Mulitplatforms 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, and they are described in “Viewing workload statistics and histograms” on page 296.

For more technical information, as well as hints and tips for WLM monitoring with PE, see “DB2 Performance Expert technical information” on page 303.

Note: At the time of writing, PE V3.1 was the latest version available, and PE V3.1 Fixpack 1 (V3.1.1) was released. PE V3.1 Fixpack 1 has additional features that enhance WLM monitoring and offer additional capabilities for monitoring multi-partition (DPF) environments. We did not rewrite this whole chapter to reflect the new changes, but have called out some enhancements where appropriate.

Also note that some of the figures in this chapter may no longer match the screen appearance in V3.1.1.

9.2 Monitoring your DB2 environment

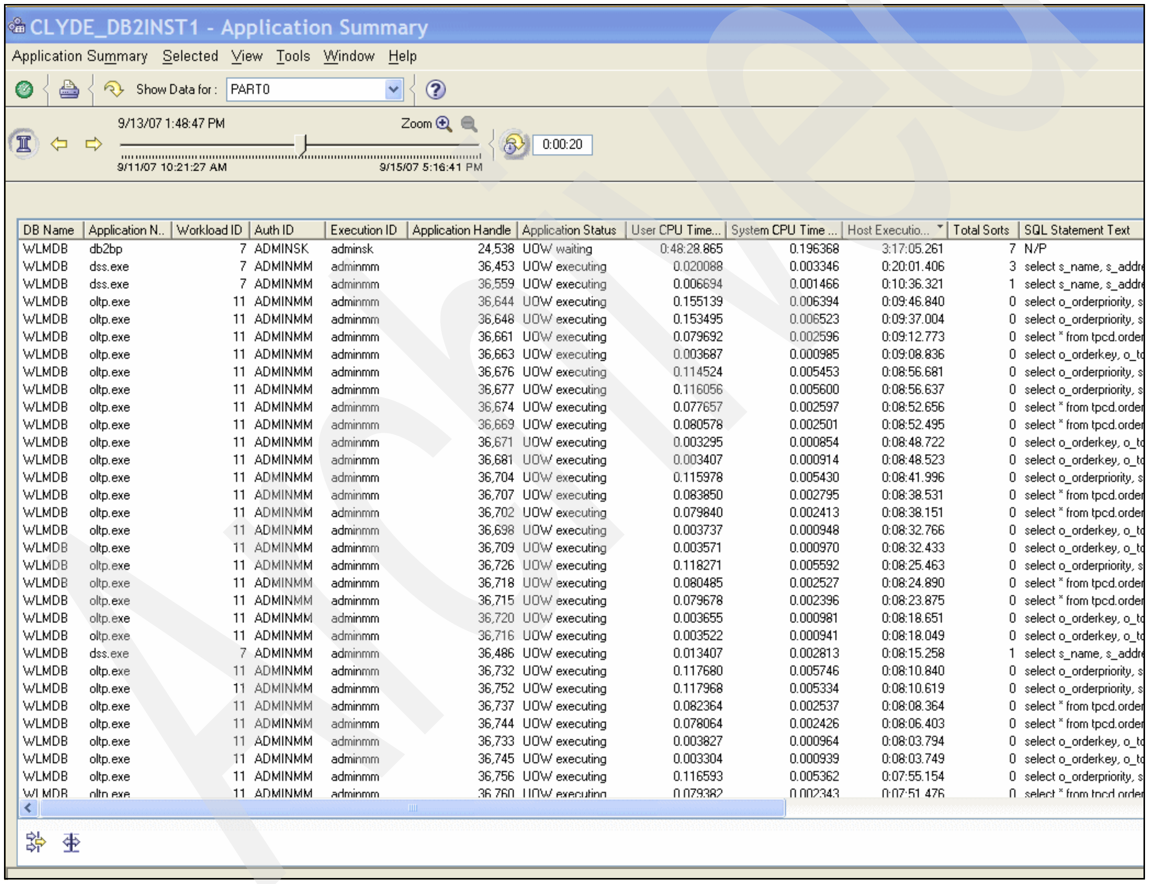
In this section, we look at a few of the basic monitoring capabilities of DB2 Performance Expert, whether you use workload management or not.

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 shown in Figure 9-1, 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
Zoom

Show Data for : PART0

9/13/07 1:48:47 PM

9/11/07 10:21:27 AM

9/15/07 5:16:41 PM

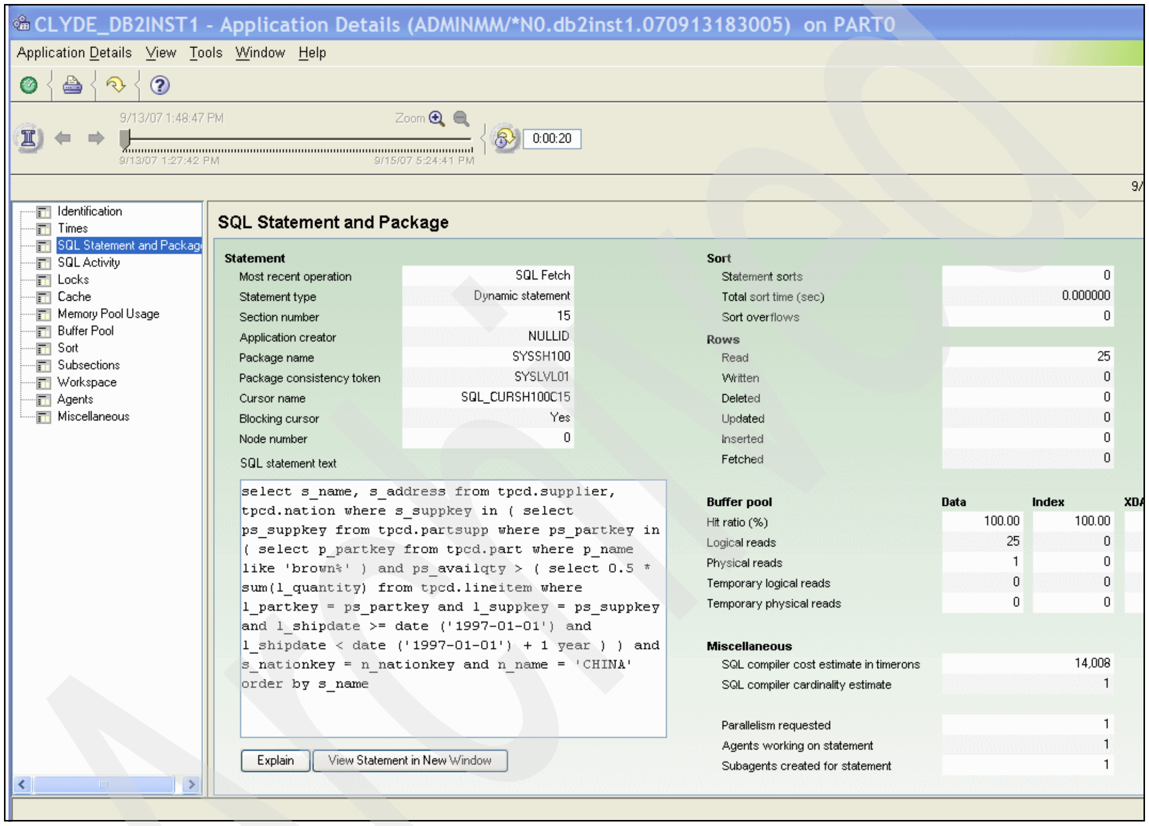
0:00:20

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_addre
WLMDB	dss.exe	7	ADMINMM	adminmm	36,559	UOW executing	0.006694	0.001466	0:10:36.321	1	select s_name, s_addre
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,644	UOW executing	0.155139	0.006394	0:09:46.840	0	select o_orderpriority, s
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,648	UOW executing	0.153495	0.006523	0:09:37.004	0	select o_orderpriority, s
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,661	UOW executing	0.079692	0.002596	0:09:12.773	0	select * from tpcd.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,663	UOW executing	0.003687	0.000985	0:09:08.836	0	select o_orderkey, o_t
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,676	UOW executing	0.114524	0.005453	0:08:56.681	0	select o_orderpriority, s
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,677	UOW executing	0.116056	0.005600	0:08:56.637	0	select o_orderpriority, s
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,674	UOW executing	0.077657	0.002597	0:08:52.656	0	select * from tpcd.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,669	UOW executing	0.080578	0.002501	0:08:52.495	0	select * from tpcd.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,671	UOW executing	0.003295	0.000854	0:08:48.722	0	select o_orderkey, o_t
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,681	UOW executing	0.003407	0.000914	0:08:48.523	0	select o_orderkey, o_t
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,704	UOW executing	0.115978	0.005430	0:08:41.996	0	select o_orderpriority, s
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,707	UOW executing	0.083850	0.002795	0:08:38.531	0	select * from tpcd.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,702	UOW executing	0.079840	0.002413	0:08:38.151	0	select * from tpcd.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,698	UOW executing	0.003737	0.000948	0:08:32.766	0	select o_orderkey, o_t
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,709	UOW executing	0.003571	0.000970	0:08:32.433	0	select o_orderkey, o_t
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,726	UOW executing	0.118271	0.005592	0:08:25.463	0	select o_orderpriority, s
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,718	UOW executing	0.080485	0.002527	0:08:24.890	0	select * from tpcd.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,715	UOW executing	0.079678	0.002396	0:08:23.875	0	select * from tpcd.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,720	UOW executing	0.003655	0.000981	0:08:18.651	0	select o_orderkey, o_t
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,716	UOW executing	0.003522	0.000941	0:08:18.049	0	select o_orderkey, o_t
WLMDB	dss.exe	7	ADMINMM	adminmm	36,486	UOW executing	0.013407	0.002813	0:08:15.258	1	select s_name, s_addre
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,732	UOW executing	0.117680	0.005746	0:08:10.840	0	select o_orderpriority, s
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,752	UOW executing	0.117968	0.005334	0:08:10.619	0	select o_orderpriority, s
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,737	UOW executing	0.082364	0.002537	0:08:08.364	0	select * from tpcd.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,744	UOW executing	0.078064	0.002426	0:08:06.403	0	select * from tpcd.order
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,733	UOW executing	0.003827	0.000964	0:08:03.794	0	select o_orderkey, o_t
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,745	UOW executing	0.003304	0.000939	0:08:03.749	0	select o_orderkey, o_t
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,756	UOW executing	0.116593	0.005362	0:07:55.154	0	select o_orderpriority, s
WLMDB	oltp.exe	11	ADMINMM	adminmm	36,760	UOW executing	0.079382	0.002343	0:07:51.476	0	select * from tpcd.order

Figure 9-1 Application Summary page

Application Details

To view details about any one application, double-click the application row in the Application Summary view, and a new window called Application Details will open, as shown in Figure 9-2. You can look at the currently executing statement, and even launch the DB2 Visual Explain. Some performance counters which are especially relevant to workload management functions are the timerons and cardinality estimates.



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
Fetched	0

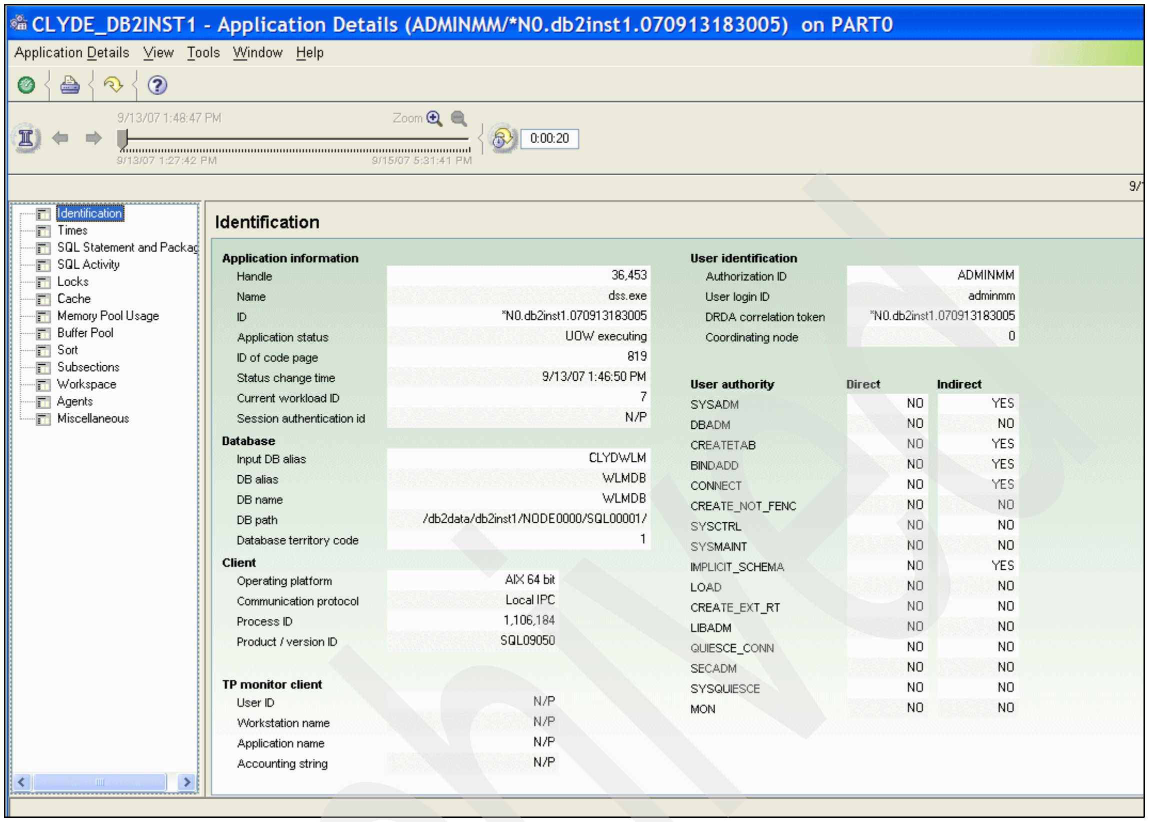
Buffer pool	**Data**	**Index**	**XDA**
Hit ratio (%)	100.00	100.00	
Logical reads	25	0	
Physical reads	1	0	
Temporary logical reads	0	0	
Temporary physical reads	0	0	

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

Figure 9-2 Application Details - SQL Statement and Package page

The Identification page of Application Details, shown in Figure 9-3, 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 3.3.2, “Creating the workloads” on page 57. In Figure 9-3 on page 276, however, 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	UOW 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 9-3 Application Details - Identification page

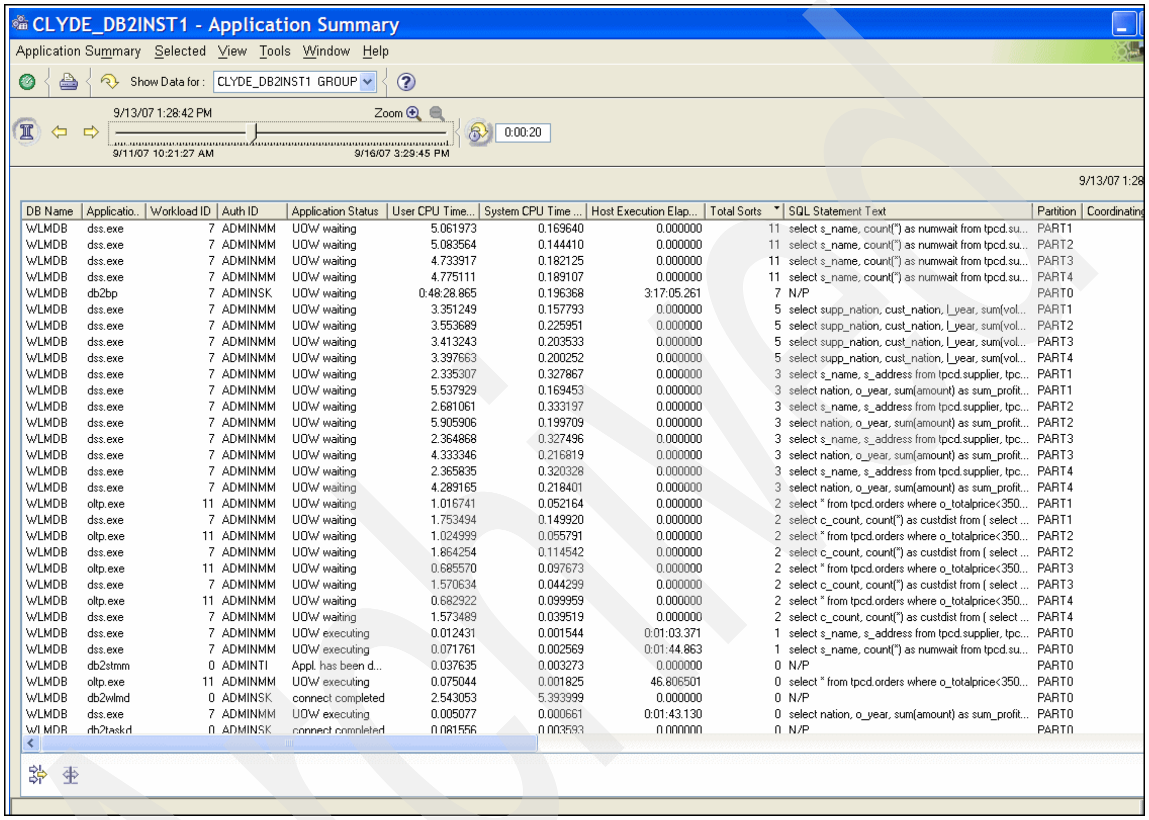
DPF considerations

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 drop-down 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.
  (This is not related to the DB2 database partition group definitions.)
- GLOBAL view - shows aggregated data from all partitions (uses GLOBAL snapshot).

Note: With DB2 Performance Expert V3.1 Fixpack 1, you also have the option of customizing the groups of partitions for monitoring. This is called a Partition Set. The Partition Sets will also appear in the drop-down list.

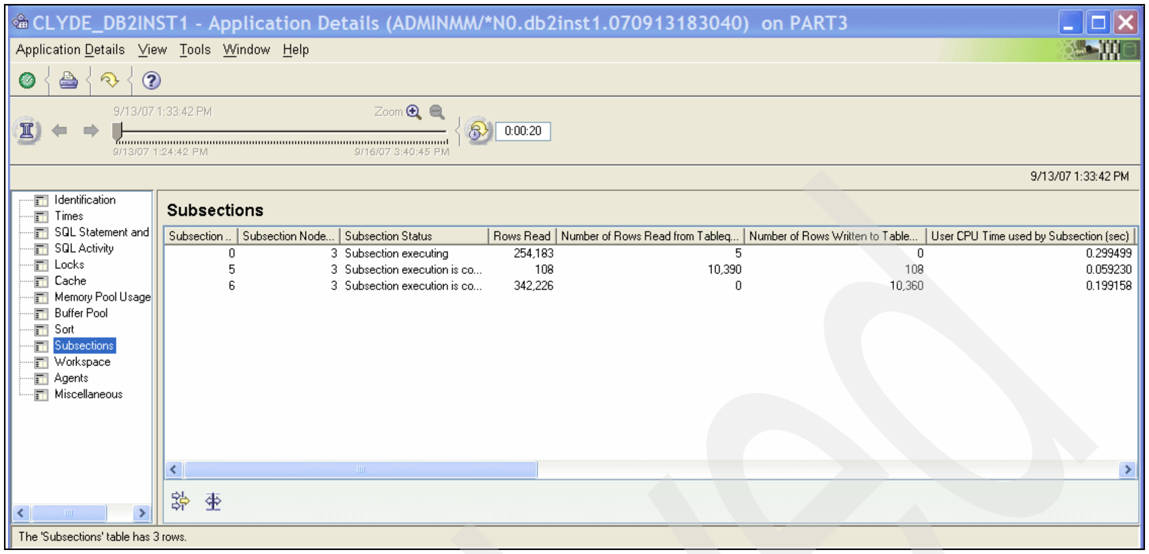
In Figure 9-4, we see an example of the GROUP view of all applications, sorted by the Total Sorts column, showing the most sorts at the top. The Group view is useful for comparing performance counters between partitions at a glance. You can display and sort different performance columns quickly see if there are skews.


CLYDE_DB2INST1 - Application Summary
Application Summary Selected View Tools Window Help
Show Data for : CLYDE_DB2INST1 GROUP
9/13/07 1:28:42 PM
Zoom
9/11/07 10:21:27 AM
9/16/07 3:29:45 PM
0:00:20
9/13/07 1:28

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_profit...	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	0.199709	0.000000	3	select nation, o_year, sum(amount) as sum_profit...	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_profit...	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_profit...	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	dss.exe	7	ADMINMM	UOW waiting	1.753494	0.149920	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	dss.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	db2wlmd	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_profit...	PART0	
WLMDB	db2taskd	0	ADMINSK	connect completed	0.081556	0.003593	0.000000	0	N/P	PART0	

Figure 9-4 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 9-5 shows an example.



Subsections

Subsection ..	Subsection Node...	Subsection Status	Rows Read	Number of Rows Read from Tableq...	Number of Rows Written to Table...	User CPU Time used by Subsection (sec)
0	3	Subsection executing	254,183	5	0	0.299499
5	3	Subsection execution is co...	108	10,390	108	0.059230
6	3	Subsection execution is co...	342,226	0	10,360	0.199158

The 'Subsections' table has 3 rows.

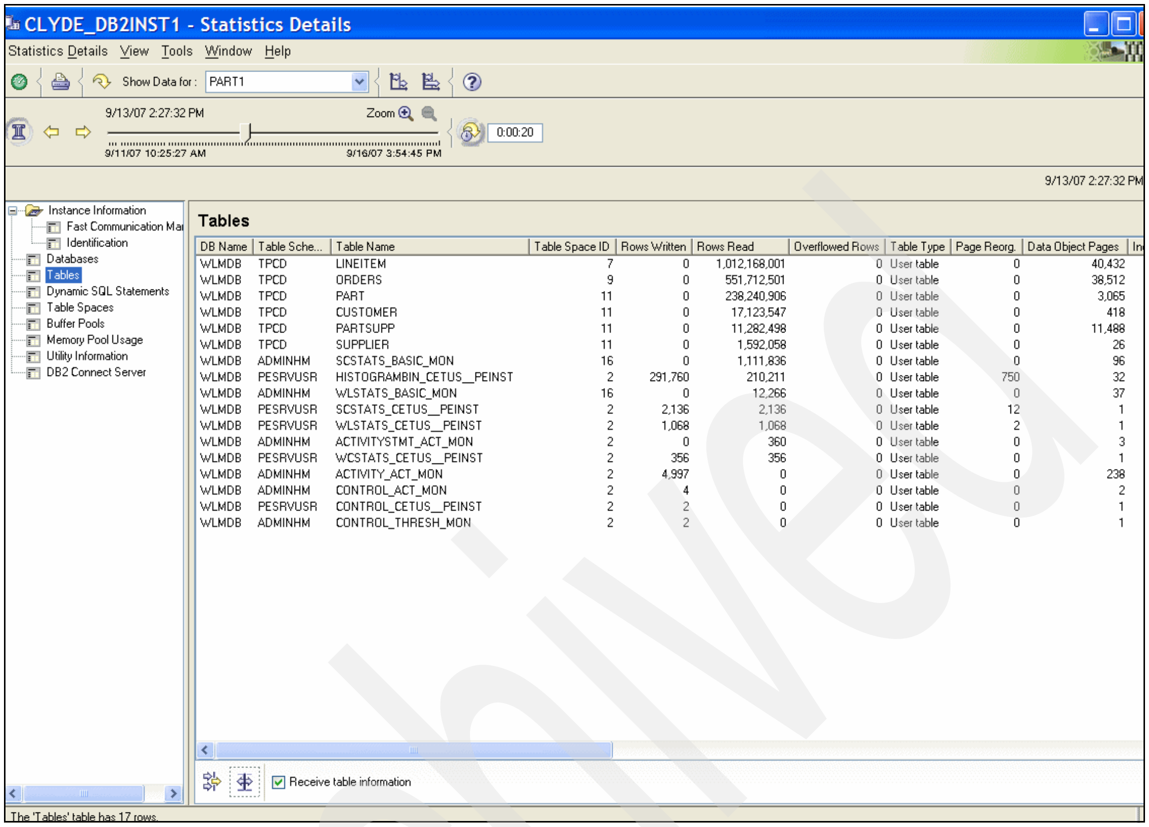
Figure 9-5 Application Details - DPF - Subsections

9.2.1 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 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 9-6, 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.



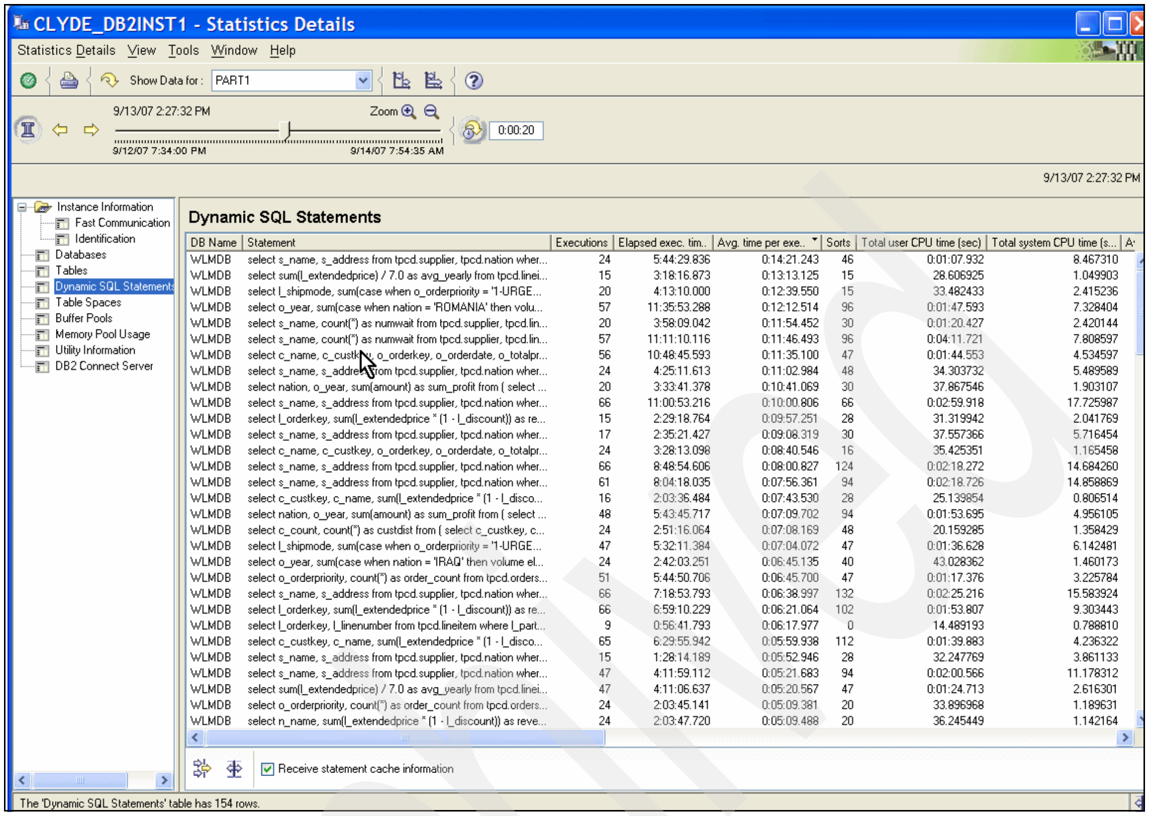
Tables

DB Name	Table Sche...	Table Name	Table Space ID	Rows Written	Rows Read	Overflowed Rows	Table Type	Page Reorg.	Data Object Pages
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

Figure 9-6 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 9-7, we see the Dynamic SQL page for Partition 1, sorted by the Average time per execution, which shows the longest-running statements at the top.



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_linenumber 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	

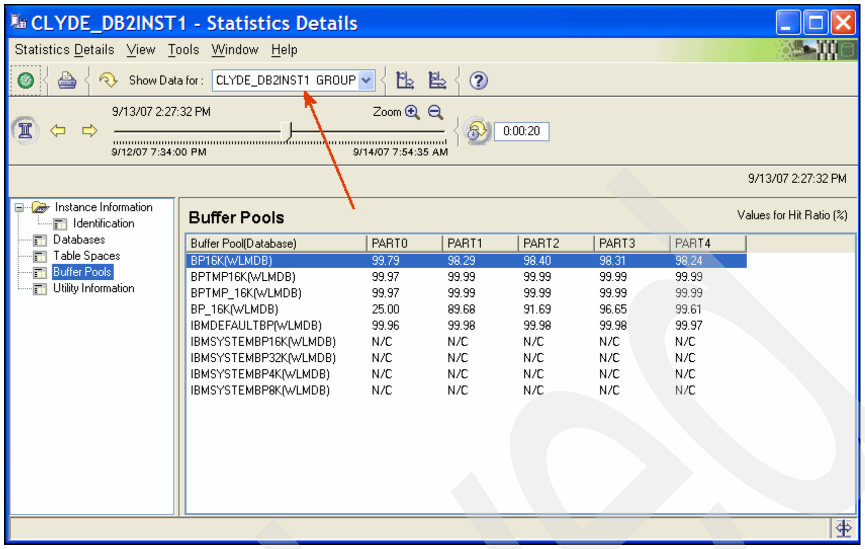
✓ Receive statement cache information

The 'Dynamic SQL Statements' table has 154 rows.

Figure 9-7 Dynamic SQL - sorted by Average time per execution - Partition 1

Buffer pool hit ratio

In Figure 9-8, 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.



Buffer Pools

Values for Hit Ratio (%)

Buffer Pool(Database)	PART0	PART1	PART2	PART3	PART4
BP16K(WLMDB)	99.79	98.29	98.40	98.31	98.24
BPTMP16K(WLMDB)	99.97	99.99	99.99	99.99	99.99
BPTMP_16K(WLMDB)	99.97	99.99	99.99	99.99	99.99
BP_16K(WLMDB)	25.00	89.68	91.69	96.65	99.61
IBMDEFAULTBP(WLMDB)	99.96	99.98	99.98	99.98	99.97
IBMSYSTEMBP16K(WLMDB)	N/C	N/C	N/C	N/C	N/C
IBMSYSTEMBP32K(WLMDB)	N/C	N/C	N/C	N/C	N/C
IBMSYSTEMBP4K(WLMDB)	N/C	N/C	N/C	N/C	N/C
IBMSYSTEMBP8K(WLMDB)	N/C	N/C	N/C	N/C	N/C

Figure 9-8 Statistics Details - Buffer pool hit ratio - Group view

9.3 Monitoring DB2 Workload Manager

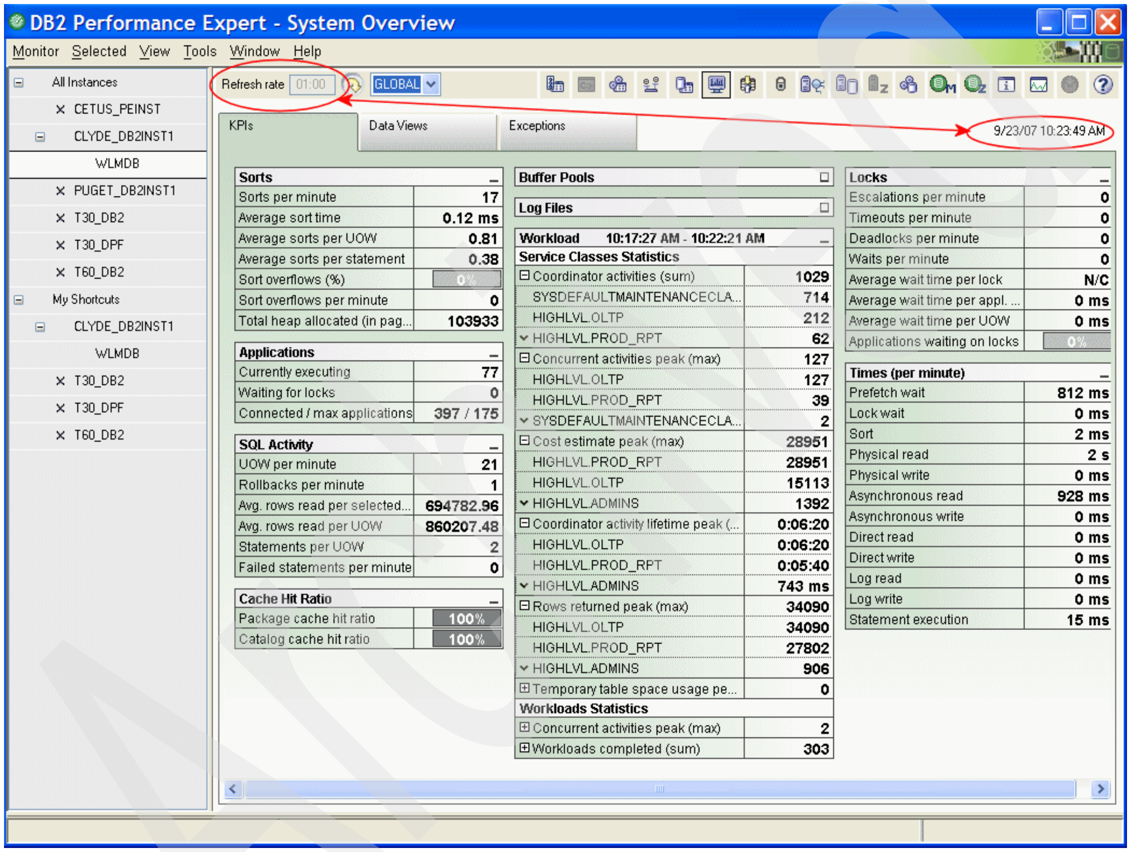
DB2 Performance Expert introduces new monitoring capabilities to coincide with the DB2 9.5 workload management features. We describe those capabilities in this section.

PE V3.1 uses only the Statistics 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. PE retrieves the data into the 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.

Note: With Fixpack1, PE now also offers the capability to create and run ACTIVITY event monitors. The event monitor data is captured into the PE tables and you can view a report or run queries against the collected data.

9.3.1 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 9-9, we see the most recent statistics captured for WLM statistics for all partitions. The counters are sorted by the “worst” at the top, which may vary by the type of counter.



Sorts	
Sorts per minute	17
Average sort time	0.12 ms
Average sorts per UOW	0.81
Average sorts per statement	0.38
Sort overflows (%)	0%
Sort overflows per minute	0
Total heap allocated (in pag...	103933

Applications	
Currently executing	77
Waiting for locks	0
Connected / max applications	397 / 175

SQL Activity	
UOW per minute	21
Rollbacks per minute	1
Avg. rows read per selected...	694782.96
Avg. rows read per UOW	860207.48
Statements per UOW	2
Failed statements per minute	0

Cache Hit Ratio	
Package cache hit ratio	100%
Catalog cache hit ratio	100%

Buffer Pools

Log Files

Workload 10:17:27 AM - 10:22:21 AM	
Service Classes Statistics	
⊟ Coordinator activities (sum)	1029
SYSDEFAULTMAINTENANCECLA...	714
HIGHLVL.OLTP	212
HIGHLVL.PROD_RPT	62
⊟ Concurrent activities peak (max)	127
HIGHLVL.OLTP	127
HIGHLVL.PROD_RPT	39
SYSDEFAULTMAINTENANCECLA...	2
⊟ Cost estimate peak (max)	28951
HIGHLVL.PROD_RPT	28951
HIGHLVL.OLTP	15113
HIGHLVL.ADMINS	1392
⊟ Coordinator activity lifetime peak (...	0:06:20
HIGHLVL.OLTP	0:06:20
HIGHLVL.PROD_RPT	0:05:40
HIGHLVL.ADMINS	743 ms
⊟ Rows returned peak (max)	34090
HIGHLVL.OLTP	34090
HIGHLVL.PROD_RPT	27802
HIGHLVL.ADMINS	906
⊞ Temporary table space usage pe...	0
Workloads Statistics	
⊞ Concurrent activities peak (max)	2
⊞ Workloads completed (sum)	303

Locks	
Escalations per minute	0
Timeouts per minute	0
Deadlocks per minute	0
Waits per minute	0
Average wait time per lock	N/C
Average wait time per appl. ...	0 ms
Average wait time per UOW	0 ms
Applications waiting on locks	0%

Times (per minute)	
Prefetch wait	812 ms
Lock wait	0 ms
Sort	2 ms
Physical read	2 s
Physical write	0 ms
Asynchronous read	928 ms
Asynchronous write	0 ms
Direct read	0 ms
Direct write	0 ms
Log read	0 ms
Log write	0 ms
Statement execution	15 ms

Figure 9-9 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 side of the window, which is controlled by a different refresh rate.

In Figure 9-9, 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 294.

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.

9.3.2 Viewing workload management definitions

To view Workload Management detail data in PE, you must click the Workload Management icon on the Toolbar on the System Overview page, as shown in Figure 9-10.

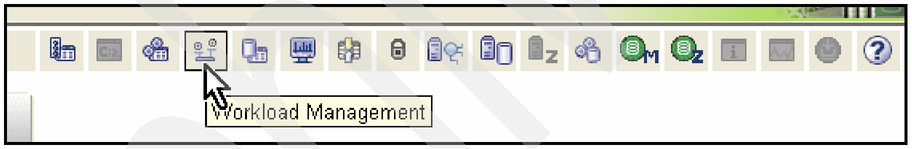
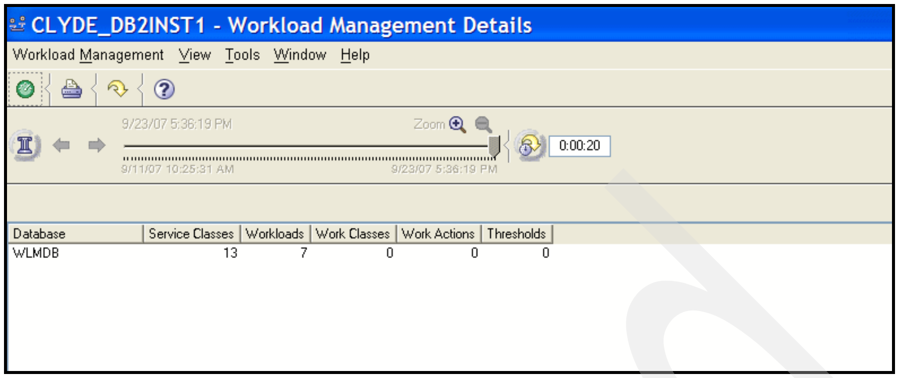


Figure 9-10 DB2 Performance Expert Toolbar

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 9-11. 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 the WLMDB.



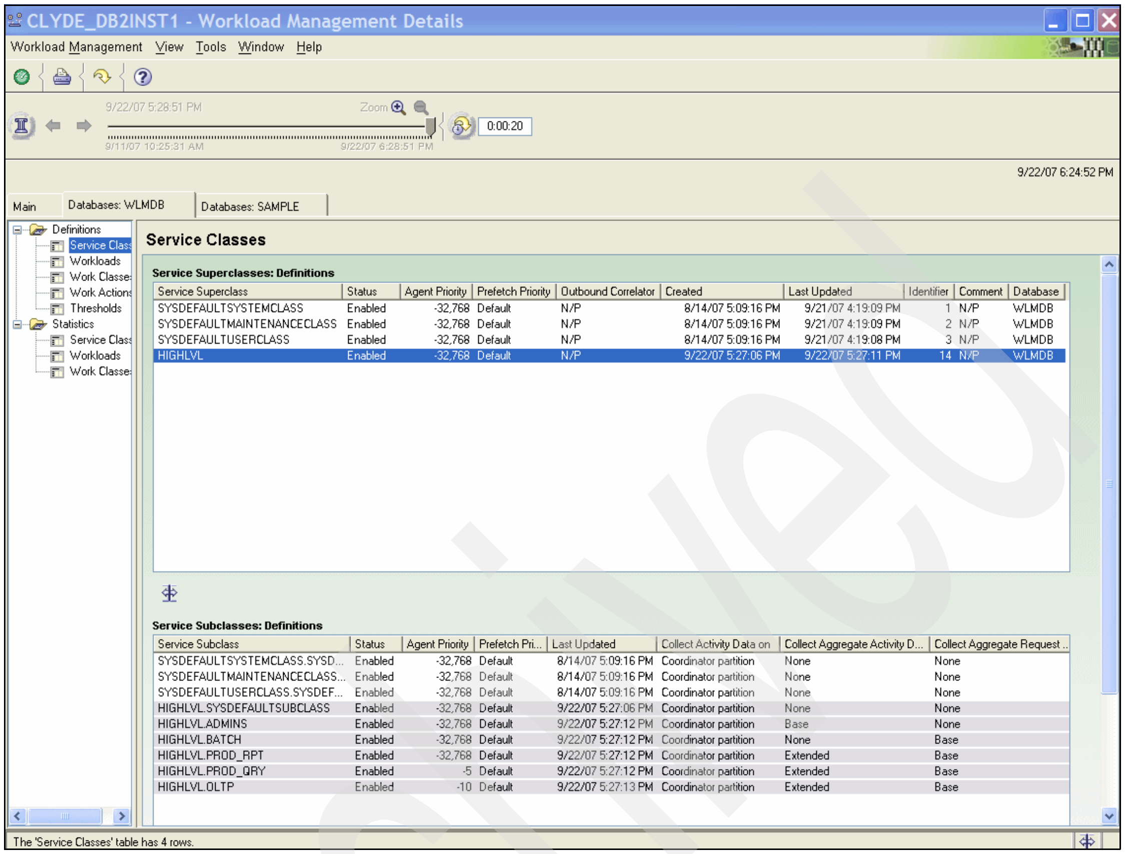
Database	Service Classes	Workloads	Work Classes	Work Actions	Thresholds
WLMDB	13	7	0	0	0

Figure 9-11   Workload Management Details

Now we will look at the same WLM definitions as described for the mixed workload in 5.3, “Manage the work” on page 126. Here we have one top-level service class with several subclasses and workloads underneath.

The definitions for the mixed workload are shown in Figure 9-12. When you select a Service Class in the upper part of the screen, its associated subclasses are highlighted in the lower part of the screen. In our mixed workload, we have several subclasses that all belong 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.



Service Classes

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

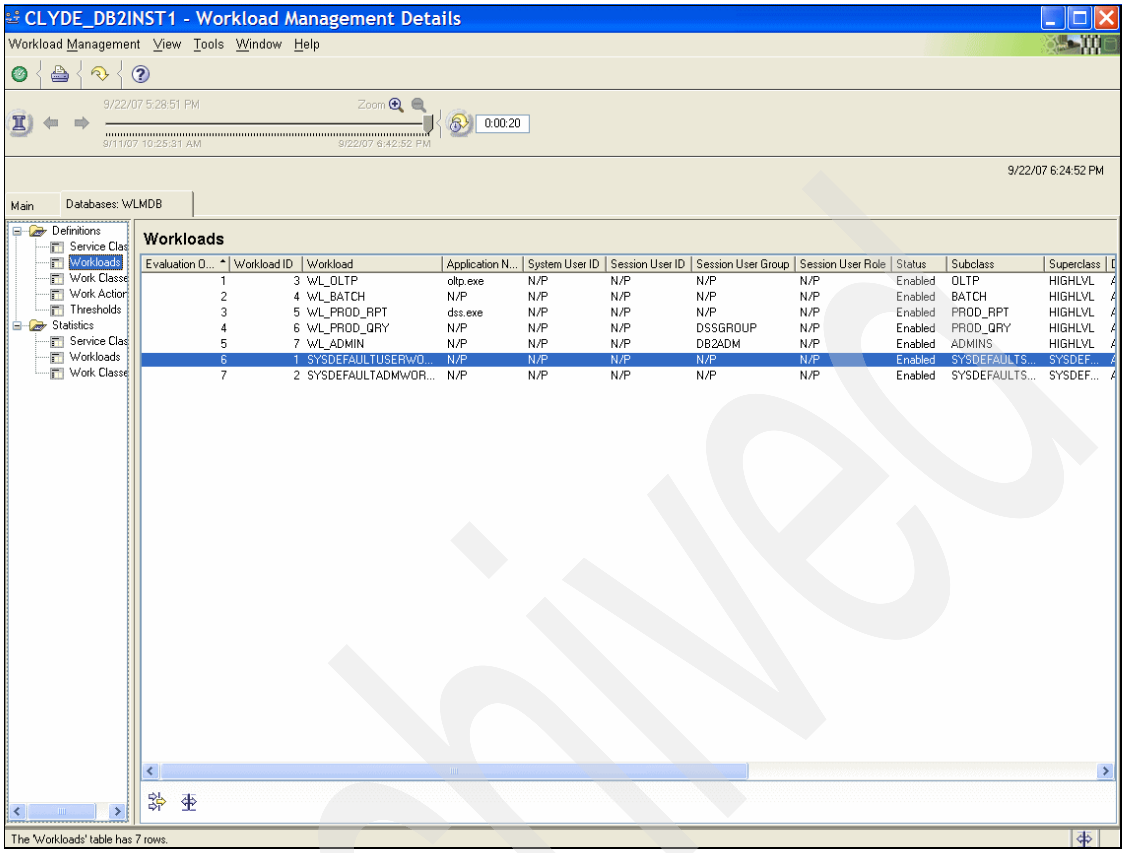
Service Subclasses: Definitions

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.SYSDEF...	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.OLTP	Enabled	-10	Default	9/22/07 5:27:13 PM	Coordinator partition	Extended	Base

The 'Service Classes' table has 4 rows.

Figure 9-12 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 9-13, we can see all the workloads for the WLMDB database, sorted by the evaluation order. In Chapter 5, “WLM sample scenarios - mixed OLTP and DSS environment” on page 123, the OLTP workload was added and placed ahead of the other workloads in the evaluation order, and indeed that is what we see here.



CLYDE_DB2INST1 - Workload Management Details

9/22/07 6:24:52 PM

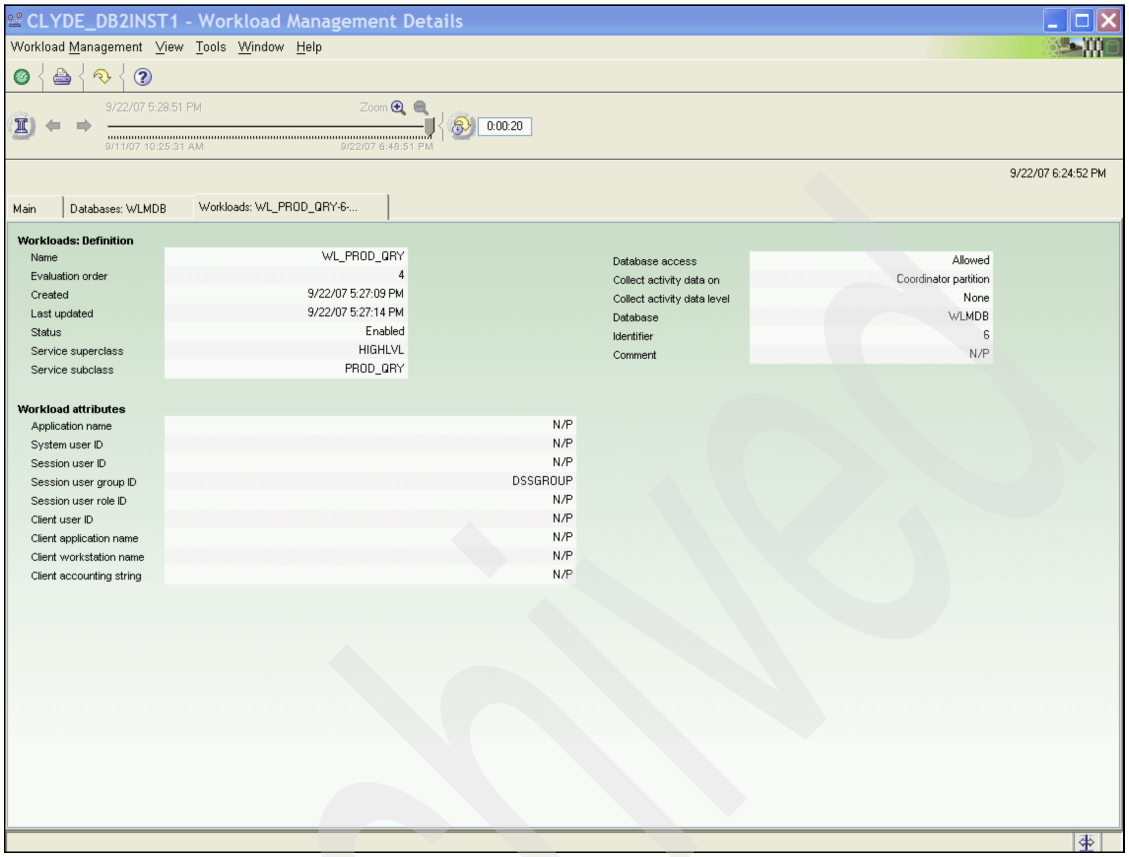
Workloads

Evaluation O...	Workload ID	Workload	Application N...	System User ID	Session User ID	Session User Group	Session User Role	Status	Subclass	Superclass	D
1	3	WL_OLTP	oltp.exe	N/P	N/P	N/P	N/P	Enabled	OLTP	HIGHLVL	A
2	4	WL_BATCH	N/P	N/P	N/P	N/P	N/P	Enabled	BATCH	HIGHLVL	A
3	5	WL_PROD_RPT	dss.exe	N/P	N/P	N/P	N/P	Enabled	PROD_RPT	HIGHLVL	A
4	6	WL_PROD_QRY	N/P	N/P	N/P	DSSGROUP	N/P	Enabled	PROD_QRY	HIGHLVL	A
5	7	WL_ADMIN	N/P	N/P	N/P	DB2ADM	N/P	Enabled	ADMINS	HIGHLVL	A
6	1	SYSDEFAULTUSERWO...	N/P	N/P	N/P	N/P	N/P	Enabled	SYSDEFAULTS...	SYSDEF...	A
7	2	SYSDEFAULTADMWOR...	N/P	N/P	N/P	N/P	N/P	Enabled	SYSDEFAULTS...	SYSDEF...	A

The 'Workloads' table has 7 rows.

Figure 9-13 WLM Workload definitions - mixed workload

To view detail about any WLM definition, double-click it. Figure 9-14 shows an example detail page for the WL_PROD_QRY workload.


CLYDE_DB2INST1 - Workload Management Details
Workload Management View Tools Window Help
9/22/07 5:28:51 PM
Zoom
0:00:20
9/11/07 10:25:31 AM
9/22/07 6:48:51 PM
9/22/07 6:24:52 PM
Main
Databases: WLMDB
Workloads: WL_PROD_QRY-6-...

Workloads: Definition

Name	WL_PROD_QRY
Evaluation order	4
Created	9/22/07 5:27:09 PM
Last updated	9/22/07 5:27:14 PM
Status	Enabled
Service superclass	HIGHLVL
Service subclass	PROD_QRY

Database access	Allowed
Collect activity data on	Coordinator partition
Collect activity data level	None
Database	WLMDB
Identifier	6
Comment	N/P

Workload attributes

Application name	N/P
System user ID	N/P
Session user ID	N/P
Session user group ID	DSSGROUP
Session user role ID	N/P
Client user ID	N/P
Client application name	N/P
Client workstation name	N/P
Client accounting string	N/P

Figure 9-14 Workload definition details

We can use Performance Expert history mode to see what the definitions were in the past.

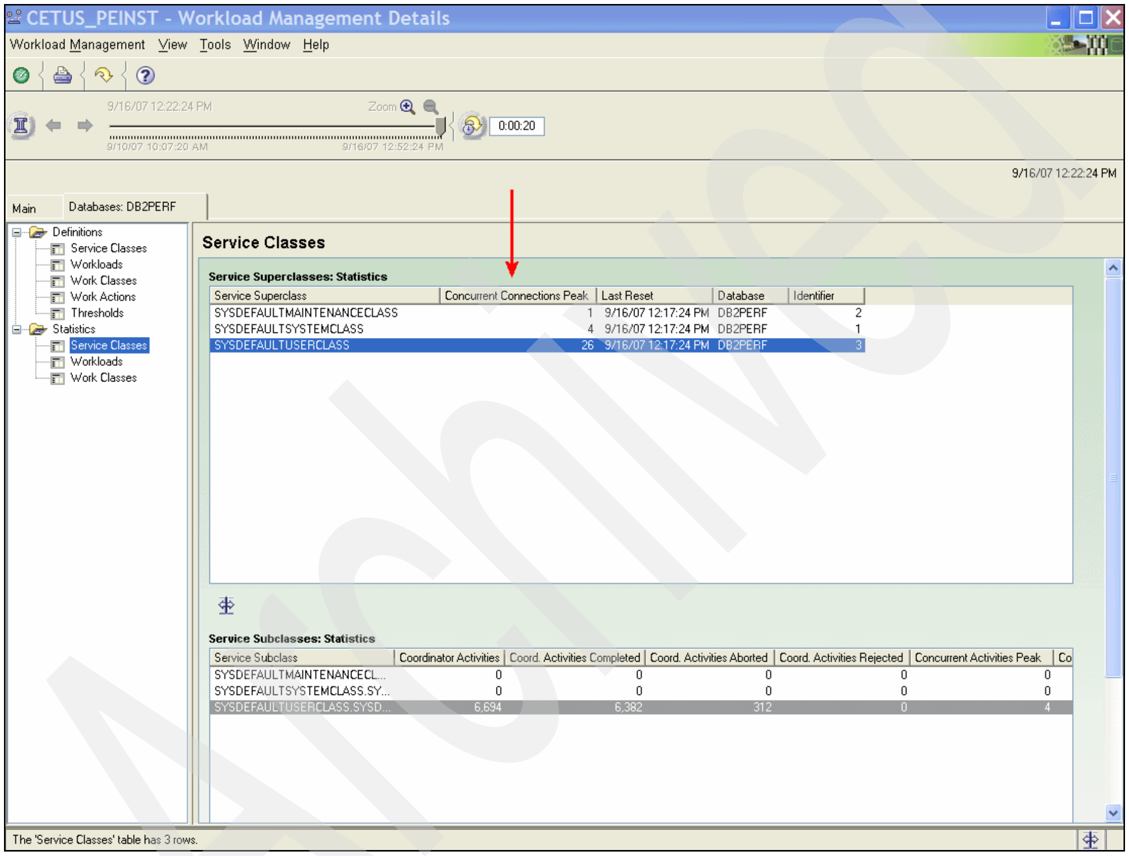
9.3.3 Viewing Workload Management statistics

Chapter 3, “Customizing the WLM execution environments” on page 49, Chapter 4, “Monitoring DB2 workload management information” on page 75, and Chapter 5, “WLM sample scenarios - mixed OLTP and DSS environment” on page 123 contain examples showing 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 “Monitoring the default WLM environment” on page 45, where you can write a query to get information. In PE, you can open the WLM Statistics page. Figure 9-15 shows using PE 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 (or subclasses) will be highlighted.



Service Superclasses: Statistics

Service Superclass	Concurrent Connections Peak	Last Reset	Database	Identifier
SYSDEFAULTMAINTENANCECLASS	1	9/16/07 12:17:24 PM	DB2PERF	2
SYSDEFAULTSYSTEMCLASS	4	9/16/07 12:17:24 PM	DB2PERF	1
SYSDEFAULTUSERCLASS	26	9/16/07 12:17:24 PM	DB2PERF	3

Service Subclasses: Statistics

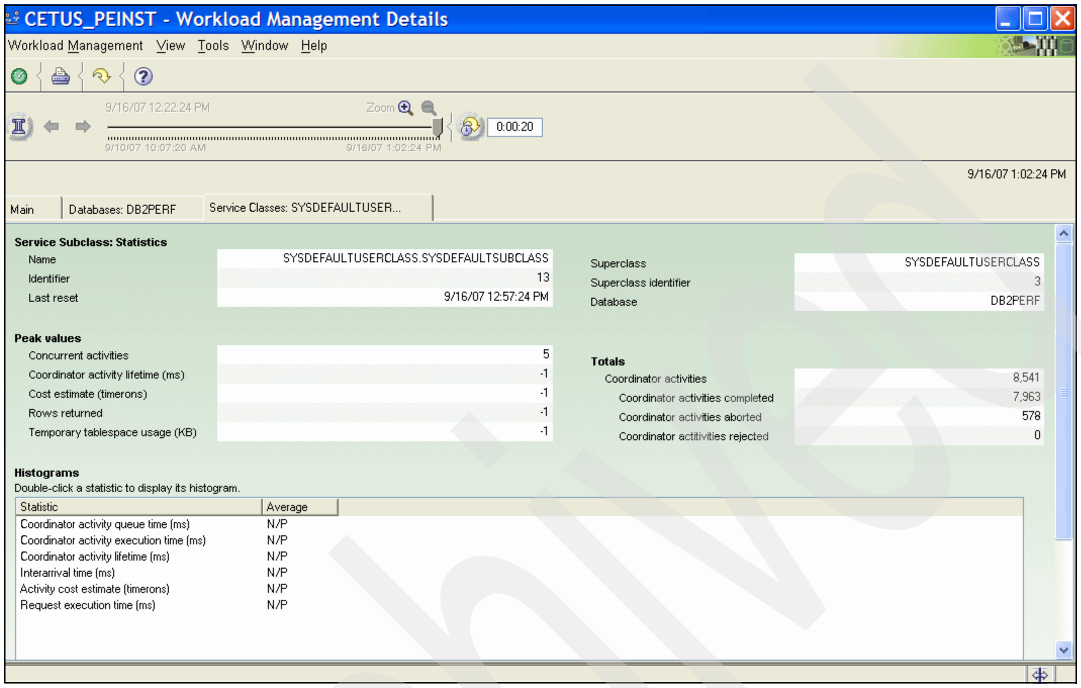
Service Subclass	Coordinator Activities	Coord. Activities Completed	Coord. Activities Aborted	Coord. Activities Rejected	Concurrent Activities Peak	Co
SYSDEFAULTMAINTENANCECL...	0	0	0	0	0	
SYSDEFAULTSYSTEMCLASS.SY...	0	0	0	0	0	
SYSDEFAULTUSERCLASS.SYSD...	6,694	6,382	312	0	4	

The 'Service Classes' table has 3 rows.

Figure 9-15 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 9-16. Here we see the same information as on the previous screen, but for the single subclass. Because 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 296.



Service Subclass: Statistics

Name	SYSDEFAULTUSERCLASS.SYSDEFAULTSUBCLASS
Identifier	13
Last reset	9/16/07 12:57:24 PM
Superclass	SYSDEFAULTUSERCLASS
Superclass identifier	3
Database	DB2PERF

Peak values

Concurrent activities	5
Coordinator activity lifetime (ms)	-1
Cost estimate (timerons)	-1
Rows returned	-1
Temporary tablespace usage (KB)	-1

Totals

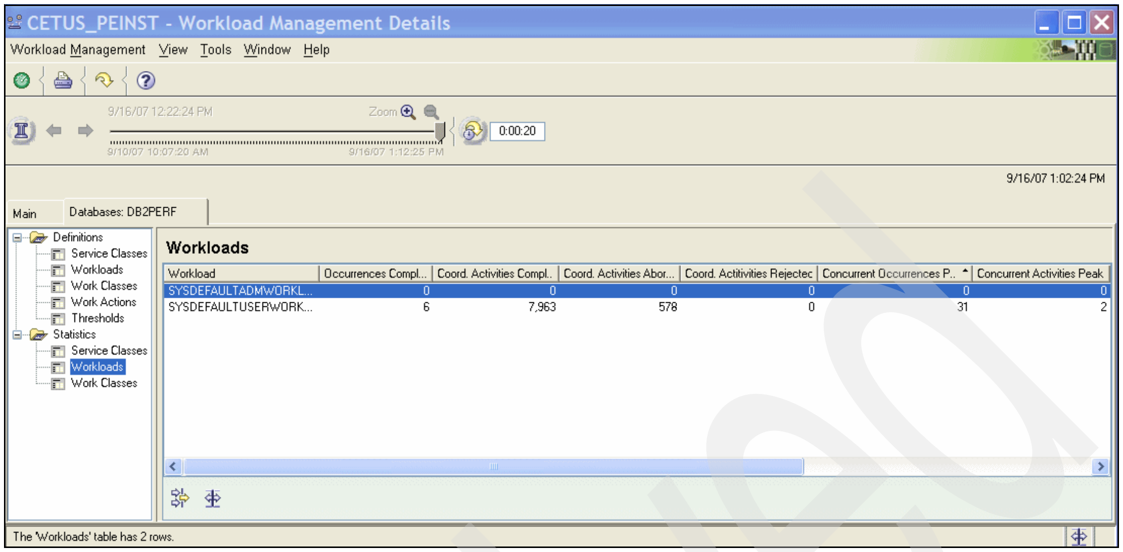
Coordinator activities	8,541
Coordinator activities completed	7,963
Coordinator activities aborted	578
Coordinator actitivities rejected	0

Histograms
Double-click a statistic to display its histogram.

Statistic	Average
Coordinator activity queue time (ms)	N/P
Coordinator activity execution time (ms)	N/P
Coordinator activity lifetime (ms)	N/P
Interarrival time (ms)	N/P
Activity cost estimate (timerons)	N/P
Request execution time (ms)	N/P

Figure 9-16 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 9-17.



Workloads

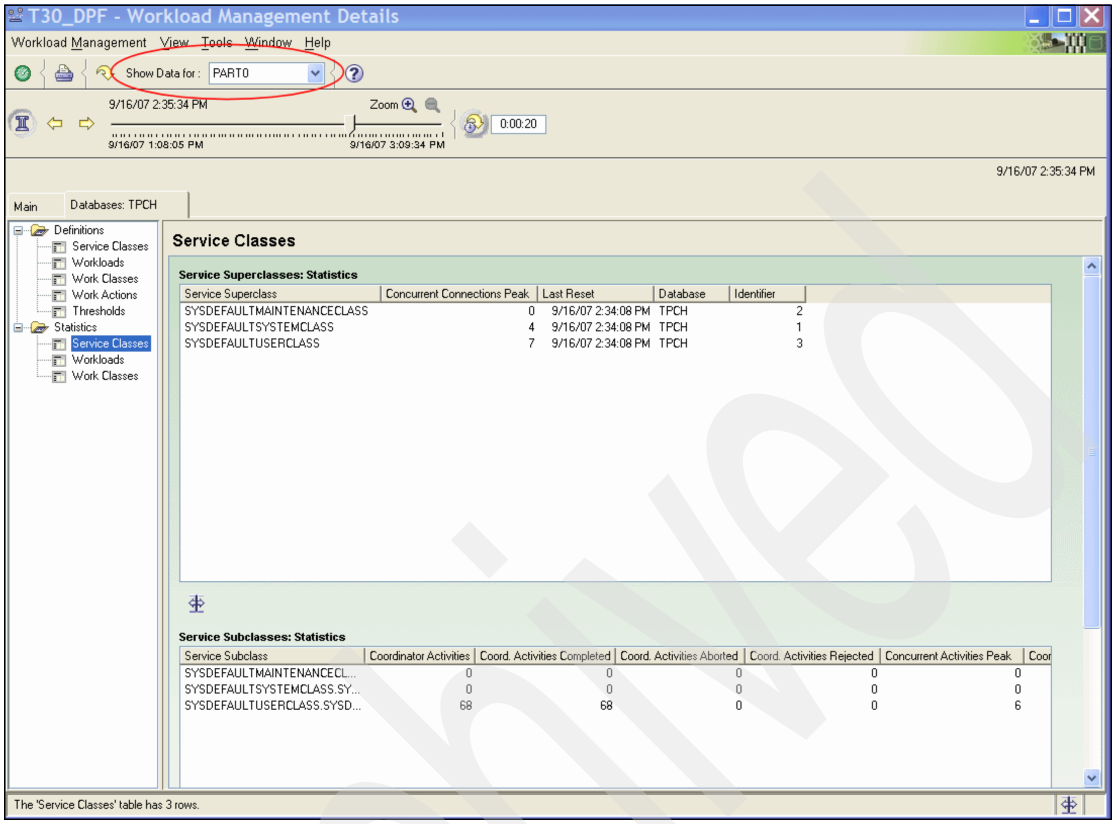
Workload	Occurrences Compl...	Coord. Activities Compl..	Coord. Activities Abor...	Coord. Actitivities Rejectec	Concurrent Occurrences P..	Concurrent Activities Peak
SYSDEFAULTADMWORKL...	0	0	0	0	0	0
SYSDEFAULTUSERWORK...	6	7,963	578	0	31	2

The 'Workloads' table has 2 rows.

Figure 9-17 WLM default Workload statistics

DPF Mode

When you are using a DPF system, you have 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 9-18, we see the same type of information as we saw in Figure 9-15 on page 288, but instead this is showing counts only for the designated partition - PART0.



Service Classes

Service Superclasses: Statistics

Service Superclass	Concurrent Connections Peak	Last Reset	Database	Identifier
SYSDEFAULTMAINTENANCECLASS	0	9/16/07 2:34:08 PM	TPCH	2
SYSDEFAULTSYSTEMCLASS	4	9/16/07 2:34:08 PM	TPCH	1
SYSDEFAULTUSERCLASS	7	9/16/07 2:34:08 PM	TPCH	3

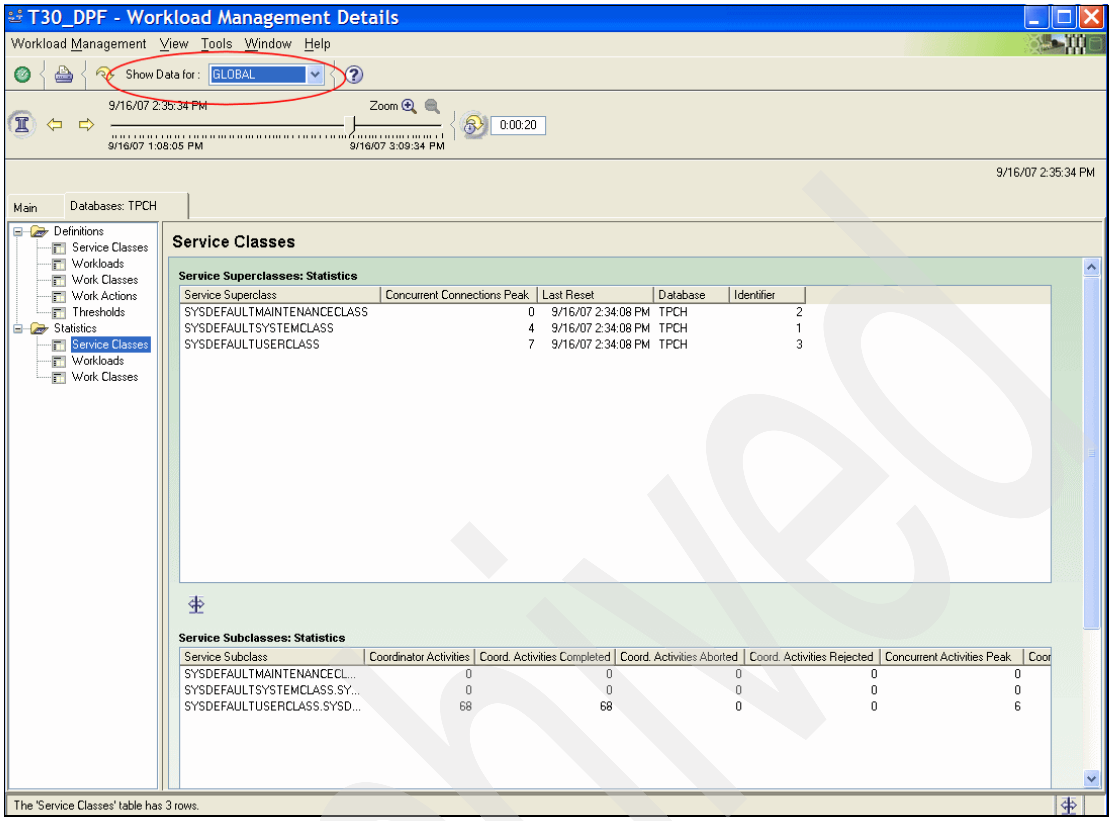
Service Subclasses: Statistics

Service Subclass	Coordinator Activities	Coord. Activities Completed	Coord. Activities Aborted	Coord. Activities Rejected	Concurrent Activities Peak	Coor
SYSDEFAULTMAINTENANCECL...	0	0	0	0	0	
SYSDEFAULTSYSTEMCLASS.SY...	0	0	0	0	0	
SYSDEFAULTUSERCLASS.SYSD...	68	68	0	0	6	

The 'Service Classes' table has 3 rows.

Figure 9-18   WLM default Service Class statistics - DPF - Partition 0

You can choose other views from the drop-down list. Next, we look at the GLOBAL view, shown in Figure 9-19. We selected GLOBAL from the list and in this case, the counts did not change.



Service Superclasses: Statistics

Service Superclass	Concurrent Connections Peak	Last Reset	Database	Identifier
SYSDEFAULTMAINTENANCECLASS	0	9/16/07 2:34:08 PM	TPCH	2
SYSDEFAULTSYSTEMCLASS	4	9/16/07 2:34:08 PM	TPCH	1
SYSDEFAULTUSERCLASS	7	9/16/07 2:34:08 PM	TPCH	3

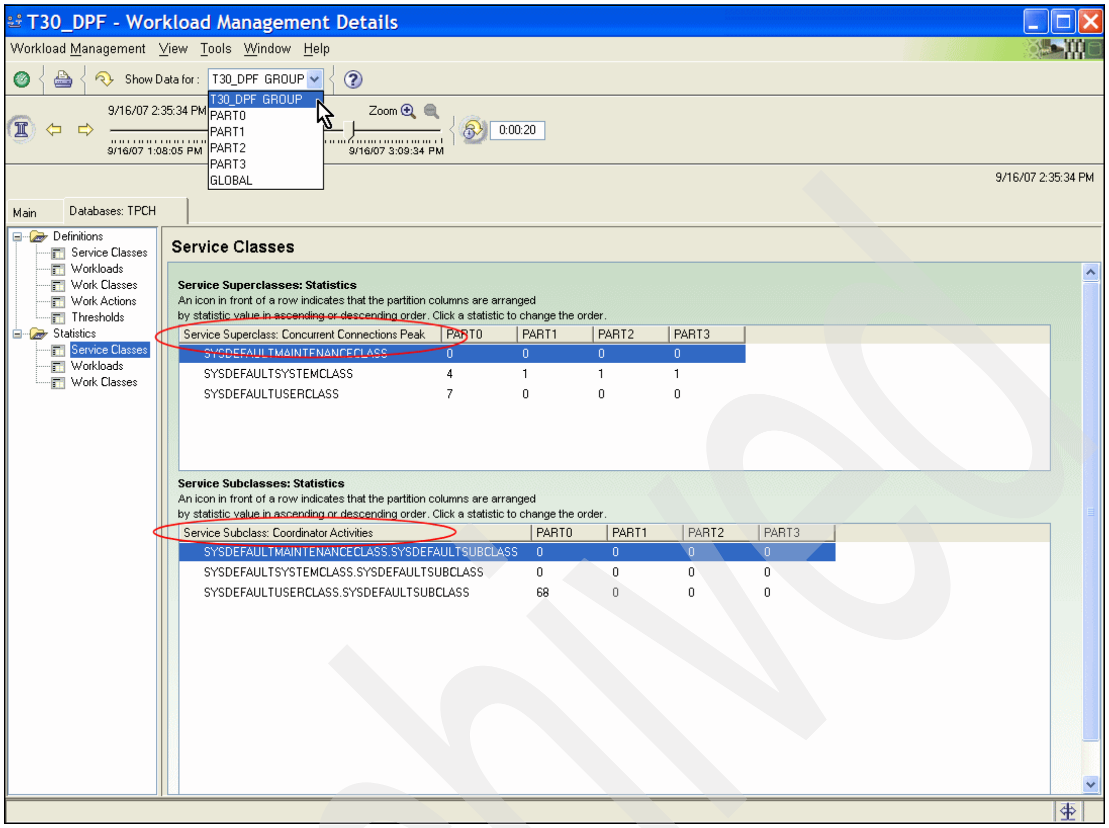
Service Subclasses: Statistics

Service Subclass	Coordinator Activities	Coord. Activities Completed	Coord. Activities Aborted	Coord. Activities Rejected	Concurrent Activities Peak	Coor
SYSDEFAULTMAINTENANCECL...	0	0	0	0	0	
SYSDEFAULTSYSTEMCLASS.SY...	0	0	0	0	0	
SYSDEFAULTUSERCLASS.SYSD...	68	68	0	0	6	

The 'Service Classes' table has 3 rows.

Figure 9-19 WLM default Service Class statistics - DPF- GLOBAL view

In Figure 9-20, we see the results of choosing the GROUP option from the drop-down list box. In this view, we can see the key counts for each partition. You cannot see all the counts for all partitions on this page, so the most critical ones are shown here.



Service Classes

Service Superclasses: Statistics
An icon in front of a row indicates that the partition columns are arranged
by statistic value in ascending or descending order. Click a statistic to change the order.

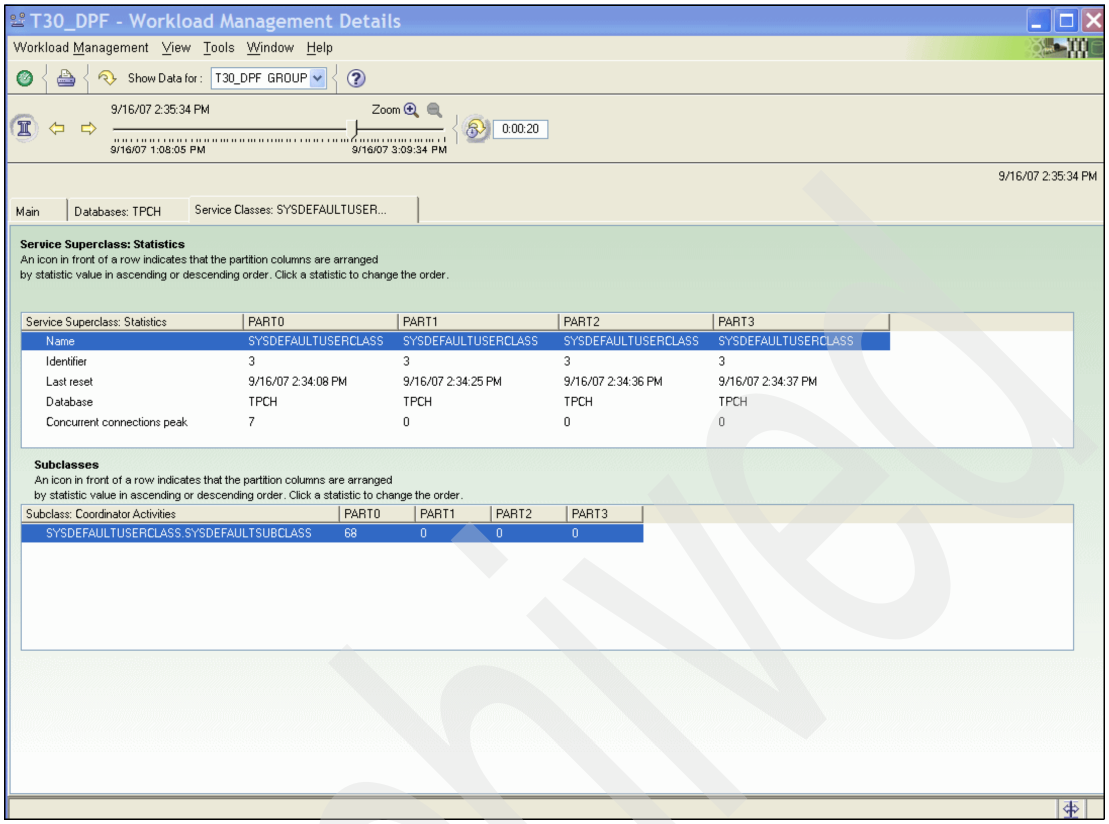
Service Superclass: Concurrent Connections Peak	PART0	PART1	PART2	PART3
SYSDEFAULTMAINTENANCECLASS	0	0	0	0
SYSDEFAULTSYSTEMCLASS	4	1	1	1
SYSDEFAULTUSERCLASS	7	0	0	0

Service Subclasses: Statistics
An icon in front of a row indicates that the partition columns are arranged
by statistic value in ascending or descending order. Click a statistic to change the order.

Service Subclass: Coordinator Activities	PART0	PART1	PART2	PART3
SYSDEFAULTMAINTENANCECLASS.SYSDEFAULTSUBCLASS	0	0	0	0
SYSDEFAULTSYSTEMCLASS.SYSDEFAULTSUBCLASS	0	0	0	0
SYSDEFAULTUSERCLASS.SYSDEFAULTSUBCLASS	68	0	0	0

Figure 9-20 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 9-21.


T30_DPF - Workload Management Details
Workload Management View Tools Window Help
Show Data for : T30_DPF GROUP
9/16/07 2:35:34 PM
Zoom
9/16/07 1:08:05 PM
9/16/07 3:09:34 PM
0:00:20
9/16/07 2:35:34 PM
Main
Databases: TPCH
Service Classes: SYSDEFAULTUSER...

Service Superclass: Statistics
An icon in front of a row indicates that the partition columns are arranged
by statistic value in ascending or descending order. Click a statistic to change the order.

Service Superclass: Statistics	PART0	PART1	PART2	PART3
Name	SYSDEFAULTUSERCLASS	SYSDEFAULTUSERCLASS	SYSDEFAULTUSERCLASS	SYSDEFAULTUSERCLASS
Identifier	3	3	3	3
Last reset	9/16/07 2:34:08 PM	9/16/07 2:34:25 PM	9/16/07 2:34:36 PM	9/16/07 2:34:37 PM
Database	TPCH	TPCH	TPCH	TPCH
Concurrent connections peak	7	0	0	0

Subclasses
An icon in front of a row indicates that the partition columns are arranged
by statistic value in ascending or descending order. Click a statistic to change the order.

Subclass: Coordinator Activities	PART0	PART1	PART2	PART3
SYSDEFAULTUSERCLASS.SYSDEFAULTSUBCLASS	68	0	0	0

Figure 9-21 WLM default Service Class statistics - DPF - GROUP details view

Monitoring non-default workloads

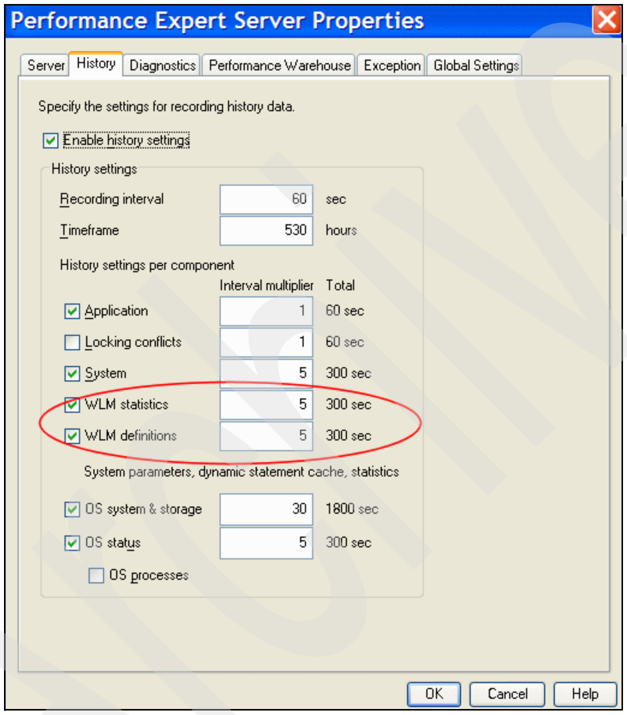
In earlier chapters of this book, we saw how to write queries against the statistics event monitor tables to get information about the workload performance. Now we explain how to do that with DB2 Performance Expert.

When you monitor the statistics with DB2 PE, you do not need to create and maintain the statistics event monitor yourself, because PE does this. You also do not need to write all your own queries to get at the information.

Assume we have a workload that is similar to the one that runs against our WLMDB database as described in 5.4.2, “Monitoring and analyzing the service classes” on page 133, or in Chapter 4, “Monitoring DB2 workload management information” on page 75. 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 zero (0), which allows PE to control its own data collection. However, 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 9-22.

> **Note:** We chose a 5-minute interval simply to hasten data collection for the purposes of demonstration. In a real-life scenario, you would choose a longer interval for both WLM statistics and WLM definition.



Specify the settings for recording history data.

☑ Enable history settings

History settings

Recording interval	60	sec
Timeframe	530	hours

History settings per component

	Interval multiplier	Total
☑ Application	1	60 sec
☐ Locking conflicts	1	60 sec
☑ System	5	300 sec
☑ WLM statistics	5	300 sec
☑ WLM definitions	5	300 sec

System parameters, dynamic statement cache, statistics

☑ OS system & storage	30	1800 sec
☑ OS status	5	300 sec
☐ OS processes		

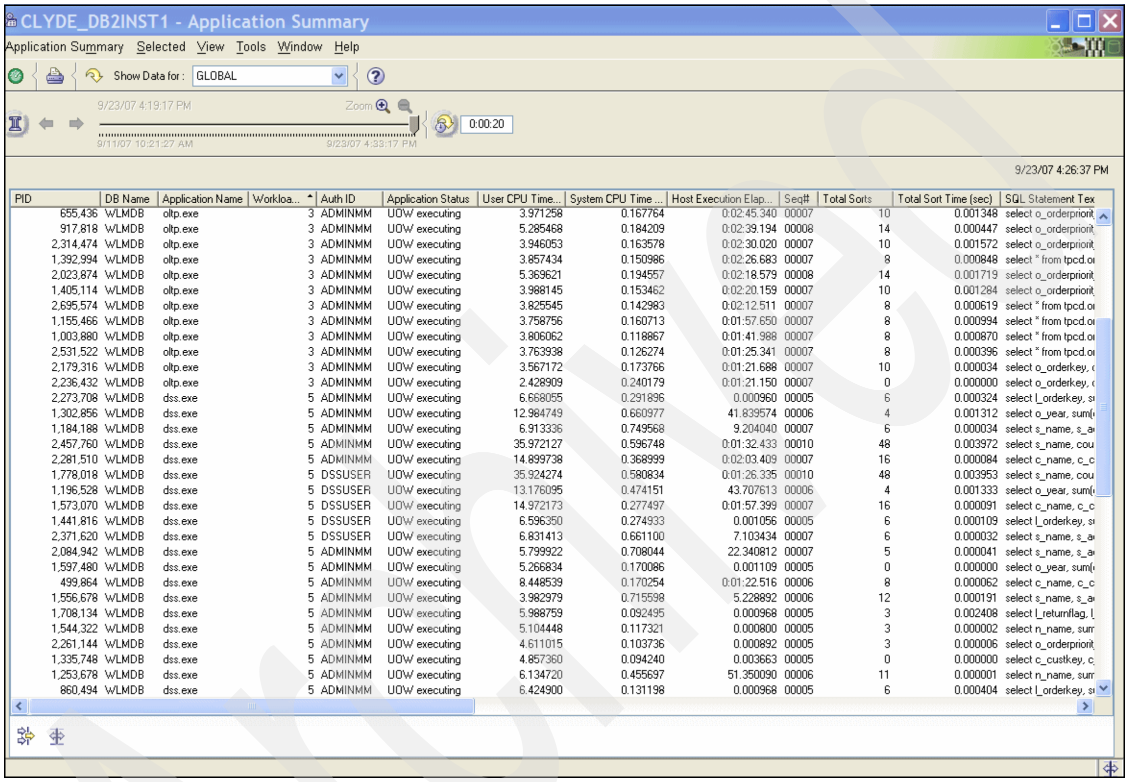
Figure 9-22 Setting the WLM statistics collection interval in DB2 Performance Expert

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 303.

The following section contains 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 9-23, 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 9-13 on page 286, that workload 3 is WL_OLTP, and workload 5 is the WL_PROD_RPT workload.



Show Data for : GLOBAL

0:00:20

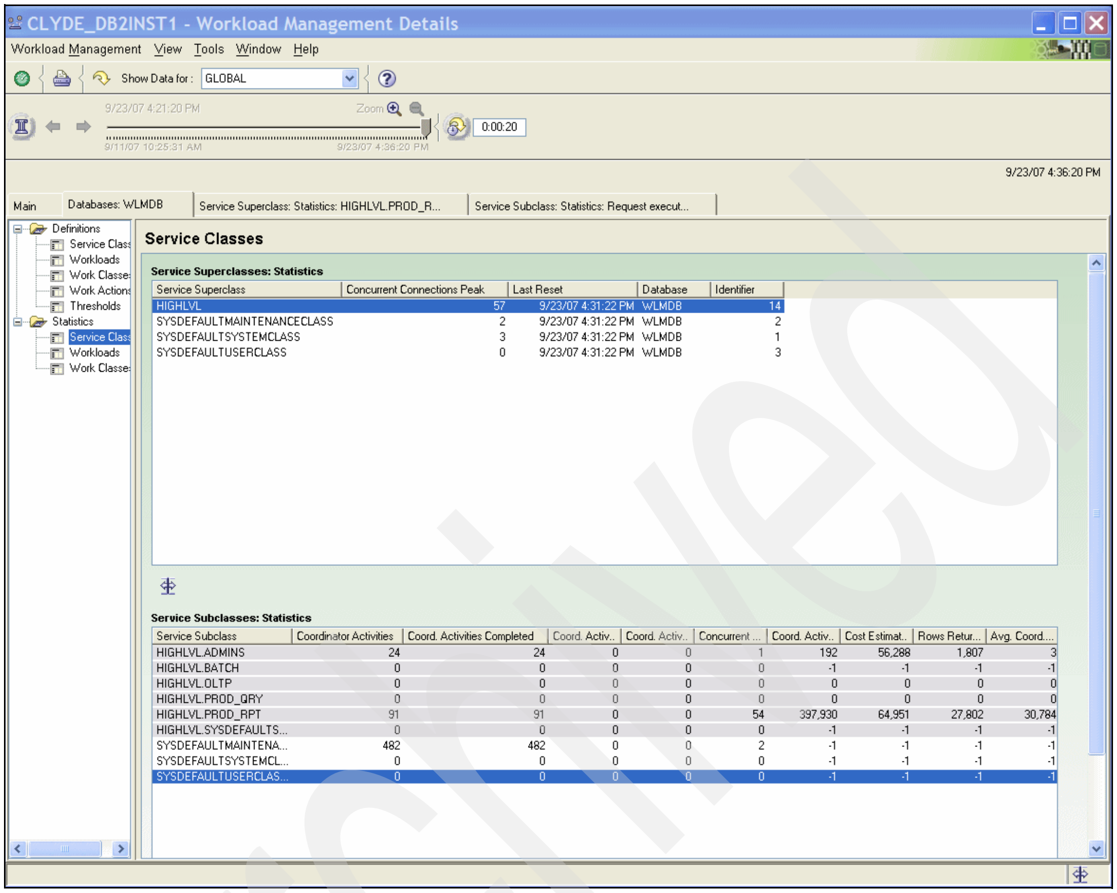
9/23/07 4:26:37 PM

PID	DB Name	Application Name	Workloa...	Auth ID	Application Status	User CPU Time...	System CPU Time ...	Host Execution Elap...	Seq#	Total Sorts	Total Sort Time (sec)	SQL Statement Tex
655,436	WLMDB	oltp.exe	3	ADMINMM	UOW executing	3.971258	0.167764	0:02:45.340	00007	10	0.001348	select o_orderpriorit
917,818	WLMDB	oltp.exe	3	ADMINMM	UOW executing	5.285468	0.184209	0:02:39.194	00008	14	0.000447	select o_orderpriorit
2,314,474	WLMDB	oltp.exe	3	ADMINMM	UOW executing	3.946053	0.163578	0:02:30.020	00007	10	0.001572	select o_orderpriorit
1,392,994	WLMDB	oltp.exe	3	ADMINMM	UOW executing	3.857434	0.150986	0:02:26.683	00007	8	0.000848	select * from tpcd.o
2,023,874	WLMDB	oltp.exe	3	ADMINMM	UOW executing	5.369621	0.194557	0:02:18.579	00008	14	0.001719	select o_orderpriorit
1,405,114	WLMDB	oltp.exe	3	ADMINMM	UOW executing	3.988145	0.153462	0:02:20.159	00007	10	0.001284	select o_orderpriorit
2,695,574	WLMDB	oltp.exe	3	ADMINMM	UOW executing	3.825545	0.142983	0:02:12.511	00007	8	0.000619	select * from tpcd.o
1,155,466	WLMDB	oltp.exe	3	ADMINMM	UOW executing	3.758756	0.160713	0:01:57.650	00007	8	0.000994	select * from tpcd.o
1,003,880	WLMDB	oltp.exe	3	ADMINMM	UOW executing	3.806062	0.118867	0:01:41.988	00007	8	0.000870	select * from tpcd.o
2,531,522	WLMDB	oltp.exe	3	ADMINMM	UOW executing	3.763938	0.126274	0:01:25.341	00007	8	0.000396	select * from tpcd.o
2,179,316	WLMDB	oltp.exe	3	ADMINMM	UOW executing	3.567172	0.173766	0:01:21.688	00007	10	0.000034	select o_orderkey,
2,236,432	WLMDB	oltp.exe	3	ADMINMM	UOW executing	2.428909	0.240179	0:01:21.150	00007	0	0.000000	select o_orderkey,
2,273,708	WLMDB	dss.exe	5	ADMINMM	UOW executing	6.668055	0.291896	0.000960	00005	6	0.000324	select l_orderkey, s
1,302,856	WLMDB	dss.exe	5	ADMINMM	UOW executing	12.984749	0.660977	41.839574	00006	4	0.001312	select o_year, sum(
1,184,188	WLMDB	dss.exe	5	ADMINMM	UOW executing	6.913336	0.749568	9.204040	00007	6	0.000034	select s_name, s_a
2,457,760	WLMDB	dss.exe	5	ADMINMM	UOW executing	35.972127	0.596748	0:01:32.433	00010	48	0.003972	select s_name, cou
2,281,510	WLMDB	dss.exe	5	ADMINMM	UOW executing	14.899738	0.368999	0:02:03.409	00007	16	0.000084	select c_name, c_c
1,778,018	WLMDB	dss.exe	5	DSSUSER	UOW executing	35.924274	0.580834	0:01:26.335	00010	48	0.003953	select s_name, cou
1,196,528	WLMDB	dss.exe	5	DSSUSER	UOW executing	13.176095	0.474151	43.707613	00006	4	0.001333	select o_year, sum(
1,573,070	WLMDB	dss.exe	5	DSSUSER	UOW executing	14.972173	0.277497	0:01:57.399	00007	16	0.000091	select c_name, c_c
1,441,816	WLMDB	dss.exe	5	DSSUSER	UOW executing	6.596350	0.274933	0.001056	00005	6	0.000109	select l_orderkey, s
2,371,620	WLMDB	dss.exe	5	DSSUSER	UOW executing	6.831413	0.661100	7.103434	00007	6	0.000032	select s_name, s_a
2,084,942	WLMDB	dss.exe	5	ADMINMM	UOW executing	5.799922	0.708044	22.340812	00007	5	0.000041	select s_name, s_a
1,597,480	WLMDB	dss.exe	5	ADMINMM	UOW executing	5.266834	0.170086	0.001109	00005	0	0.000000	select o_year, sum(
499,864	WLMDB	dss.exe	5	ADMINMM	UOW executing	8.448539	0.170254	0:01:22.516	00006	8	0.000062	select c_name, c_c
1,556,678	WLMDB	dss.exe	5	ADMINMM	UOW executing	3.982979	0.715598	5.228892	00006	12	0.000191	select s_name, s_a
1,708,134	WLMDB	dss.exe	5	ADMINMM	UOW executing	5.988759	0.092495	0.000968	00005	3	0.002408	select l_returnflag, l
1,544,322	WLMDB	dss.exe	5	ADMINMM	UOW executing	5.104448	0.117321	0.000800	00005	3	0.000002	select n_name, sum
2,261,144	WLMDB	dss.exe	5	ADMINMM	UOW executing	4.611015	0.103736	0.000892	00005	3	0.000006	select o_orderpriorit
1,335,748	WLMDB	dss.exe	5	ADMINMM	UOW executing	4.857360	0.094240	0.003663	00005	0	0.000000	select c_custkey, c
1,253,678	WLMDB	dss.exe	5	ADMINMM	UOW executing	6.134720	0.455697	51.350090	00006	11	0.000001	select n_name, sum
860,494	WLMDB	dss.exe	5	ADMINMM	UOW executing	6.424900	0.131198	0.000968	00005	6	0.000404	select l_orderkey, s

Figure 9-23 Application Summary - sorted by Workload ID

Viewing workload statistics and histograms

In Figure 9-24, we can see a summary view of all the most recently captured WLM statistics data.



Service Classes

Service Superclasses: Statistics

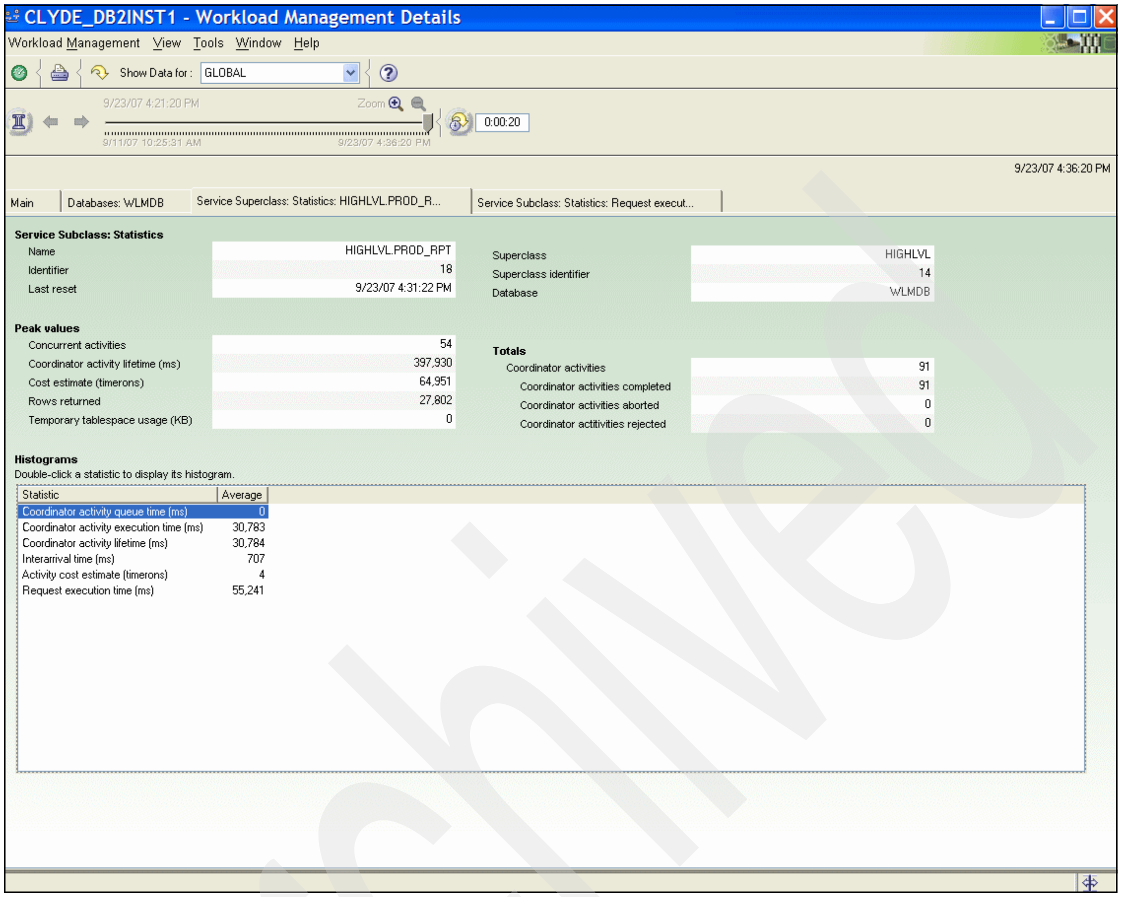
Service Superclass	Concurrent Connections Peak	Last Reset	Database	Identifier
HIGHLVL	57	9/23/07 4:31:22 PM	WLMDB	14
SYSDEFAULTMAINTENANCECLASS	2	9/23/07 4:31:22 PM	WLMDB	2
SYSDEFAULTSYSTEMCLASS	3	9/23/07 4:31:22 PM	WLMDB	1
SYSDEFAULTUSERCLASS	0	9/23/07 4:31:22 PM	WLMDB	3

Service Subclasses: Statistics

Service Subclass	Coordinator Activities	Coord. Activities Completed	Coord. Activ..	Coord. Activ..	Concurrent ...	Coord. Activ..	Cost Estimat..	Rows Retur...	Avg. Coord....
HIGHLVL.ADMINS	24	24	0	0	1	192	56,288	1,807	3
HIGHLVL.BATCH	0	0	0	0	0	-1	-1	-1	-1
HIGHLVL.OLTP	0	0	0	0	0	0	0	0	0
HIGHLVL.PROD_QRY	0	0	0	0	0	0	0	0	0
HIGHLVL.PROD_RPT	91	91	0	0	54	397,930	64,951	27,802	30,784
HIGHLVL.SYSDEFAULTS...	0	0	0	0	0	-1	-1	-1	-1
SYSDEFAULTMAINTENA...	482	482	0	0	2	-1	-1	-1	-1
SYSDEFAULTSYSTEMCL...	0	0	0	0	0	-1	-1	-1	-1
SYSDEFAULTUSERCLAS...	0	0	0	0	0	-1	-1	-1	-1

Figure 9-24 WLM Service Class statistics - summary view

The PROD_RPT service class is the only one with statistics at the moment, so we want to drill down to see more information about that one. We double-click the HIGHLVL.PROD_RPT subclass on the lower part of the window. This opens up another tab, as we see in Figure 9-25. 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.


CLYDE_DB2INST1 - Workload Management Details
Workload Management View Tools Window Help
Show Data for : GLOBAL
9/23/07 4:21:20 PM
Zoom
9/11/07 10:25:31 AM
9/23/07 4:36:20 PM
0:00:20
9/23/07 4:36:20 PM
Main
Databases: WLMDB
Service Superclass: Statistics: HIGHLVL.PROD_R...
Service Subclass: Statistics: Request execut...

Service Subclass: Statistics

Name	HIGHLVL.PROD_RPT
Identifier	18
Last reset	9/23/07 4:31:22 PM
Superclass	HIGHLVL
Superclass identifier	14
Database	WLMDB

Peak values

Concurrent activities	54
Coordinator activity lifetime (ms)	397,930
Cost estimate (timerons)	64,951
Rows returned	27,802
Temporary tablespace usage (KB)	0

Totals

Coordinator activities	91
Coordinator activities completed	91
Coordinator activities aborted	0
Coordinator actitivities rejected	0

Histograms

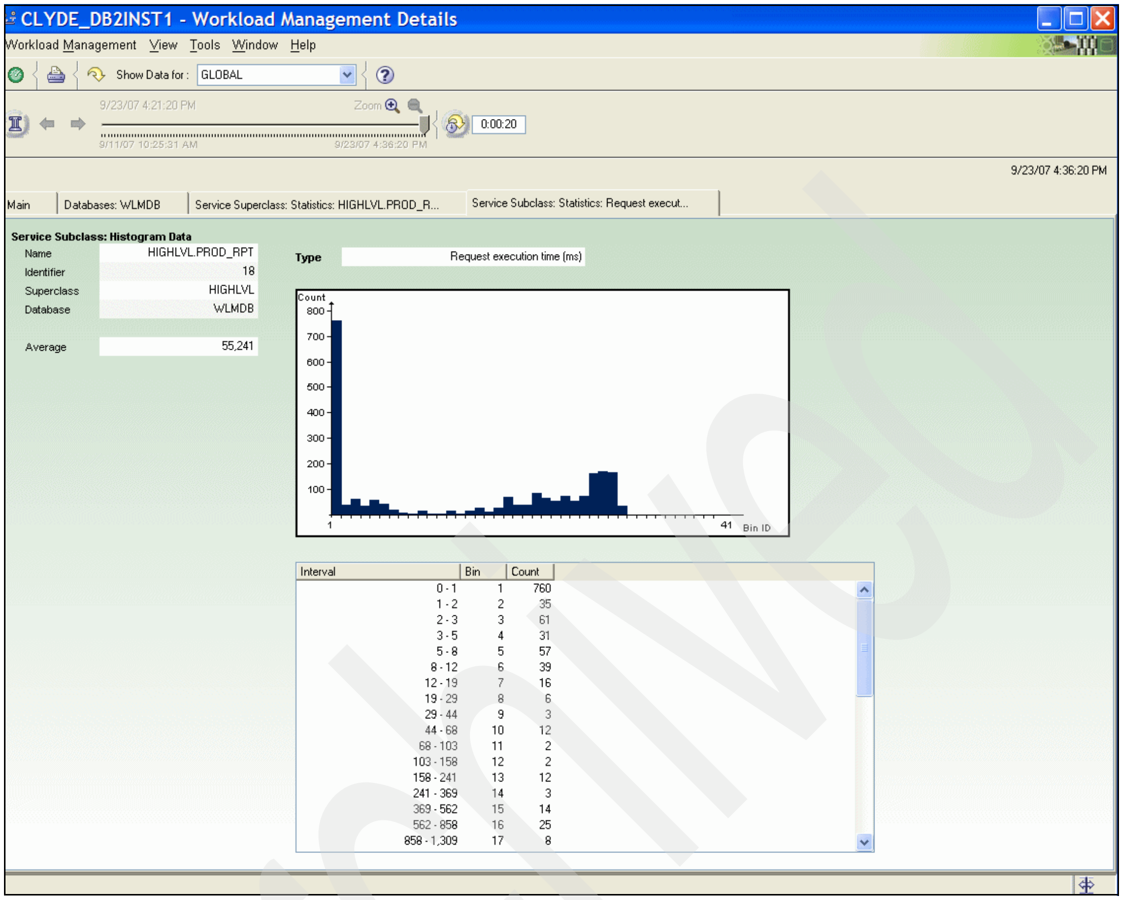
Double-click a statistic to display its histogram.

Statistic	Average
Coordinator activity queue time (ms)	0
Coordinator activity execution time (ms)	30,783
Coordinator activity lifetime (ms)	30,784
Interarrival time (ms)	707
Activity cost estimate (timerons)	4
Request execution time (ms)	55,241

Figure 9-25 WLM Subclass statistics - HIGHLVL.PROD_RPT

We double-click the Request execution time (ms) statistic, which opens up another tab where we can view the histogram chart, shown in Figure 9-26.

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.


CLYDE_DB2INST1 - Workload Management Details
Workload Management View Tools Window Help
Show Data for : GLOBAL
9/23/07 4:21:20 PM
Zoom
9/11/07 10:25:31 AM
9/23/07 4:36:20 PM
0:00:20
9/23/07 4:36:20 PM
Main
Databases: WLMDB
Service Superclass: Statistics: HIGHLVL.PROD_R...
Service Subclass: Statistics: Request execut...
Count
Bin ID

Service Subclass: Histogram Data

Name	HIGHLVL.PROD_RPT
Identifier	18
Superclass	HIGHLVL
Database	WLMDB
Average	55,241

Type: Request execution time (ms)

Interval	Bin	Count
0 - 1	1	760
1 - 2	2	35
2 - 3	3	61
3 - 5	4	31
5 - 8	5	57
8 - 12	6	39
12 - 19	7	16
19 - 29	8	6
29 - 44	9	3
44 - 68	10	12
68 - 103	11	2
103 - 158	12	2
158 - 241	13	12
241 - 369	14	3
369 - 562	15	14
562 - 858	16	25
858 - 1,309	17	8

Figure 9-26 WLM Histogram view for PROD_RPT Request execution time

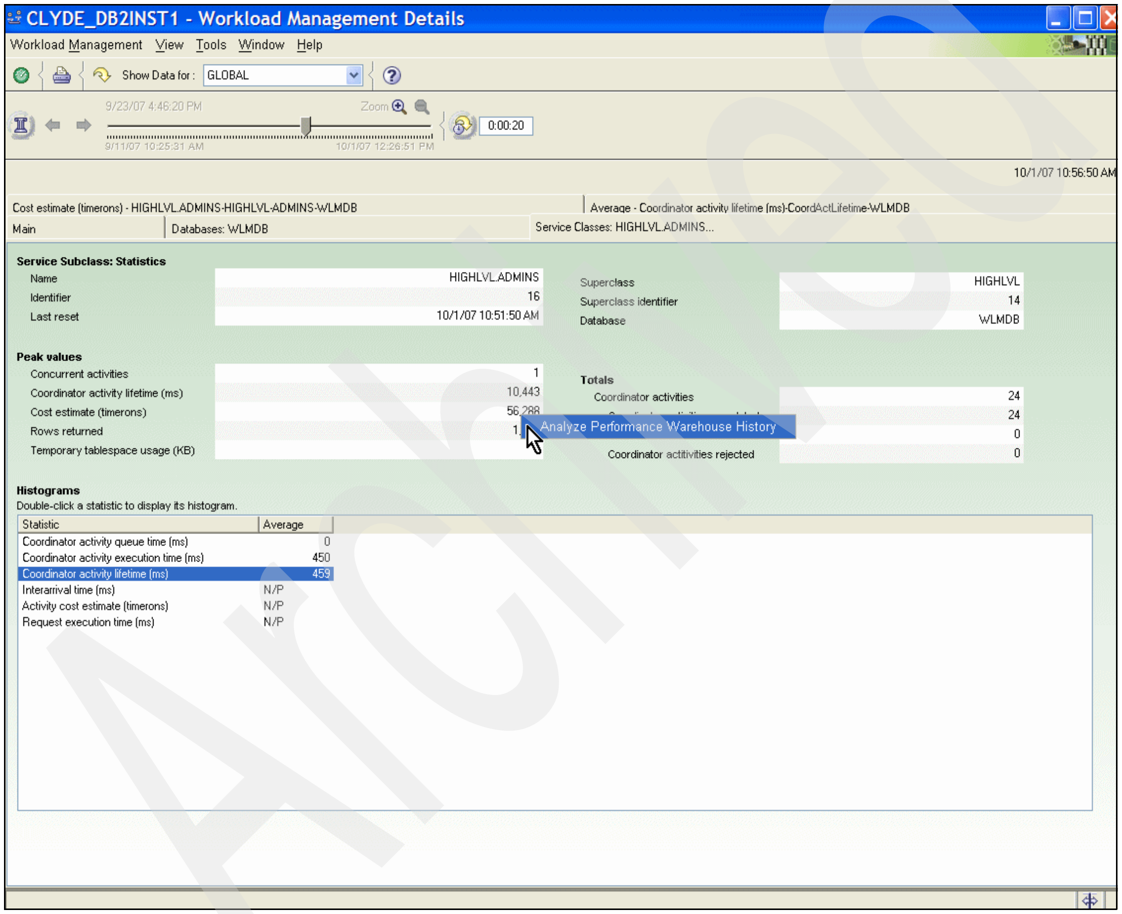
Viewing long-term WLM statistics through PE GUI

The DB2 Performance Expert server captures DB2 performance statistics using the snapshot facility, and captures WLM performance statistics using the WLM Statistics 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 operating system counters from the GUI screen, in the form of a trend analysis graph.

To access the PWH trend analysis, right-click the performance counter you are interested in, which brings up the context menu. Not all performance counters can be viewed this way, so if the context menu item is disabled (grayed out), it means that the counter is not available. If it is not grayed out, select the item as shown in Figure 9-27. In this case we are looking at a WLM statistic - the peak value for cost estimate (timerons) for the HIGHLVL.ADMINS service class.


CLYDE_DB2INST1 - Workload Management Details
Workload Management
View
Tools
Window
Help
Show Data for : GLOBAL
9/23/07 4:46:20 PM
Zoom
9/11/07 10:25:31 AM
10/1/07 12:26:51 PM
0:00:20
10/1/07 10:56:50 AM
Cost estimate (timerons) - HIGHLVL.ADMINS-HIGHLVL-ADMINS-WLMDB
Average - Coordinator activity lifetime (ms)-CoordActLifetime-WLMDB
Main
Databases: WLMDB
Service Classes: HIGHLVL.ADMINS...
Analyze Performance Warehouse History

Service Subclass: Statistics

Name	HIGHLVL.ADMINS
Identifier	16
Last reset	10/1/07 10:51:50 AM

Superclass	HIGHLVL
Superclass identifier	14
Database	WLMDB

Peak values

Concurrent activities	1
Coordinator activity lifetime (ms)	10,443
Cost estimate (timerons)	56,288
Rows returned	1
Temporary tablespace usage (KB)	

Totals

Coordinator activities	24
	24
	0
Coordinator actitivities rejected	0

Histograms

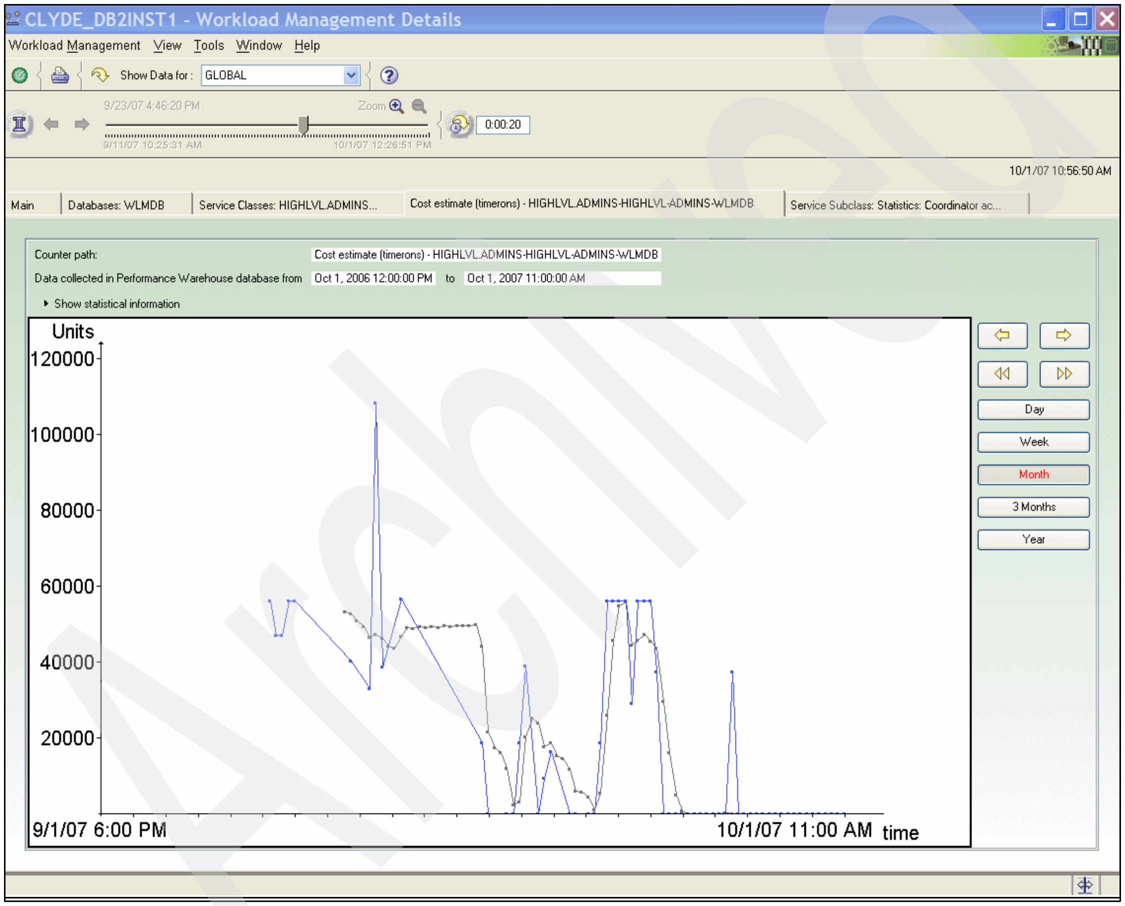
Double-click a statistic to display its histogram.

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 9-27 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 9-28, 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 in enabling you to quickly recognize when something may be trending out of good performance, or that perhaps a temporary spike has occurred that you can investigate further. Along with the WLM counters we see here, the trend charts are available for operating system and DB2 statistics counters, such as paging space usage, buffer pool hit ratios, table pages used, rows read and so on.


CLYDE_DB2INST1 - Workload Management Details
Workload Management View Tools Window Help
Zoom
Main
Databases: WLMDB
Service Classes: HIGHLVLADMINS...
Cost estimate (timerons) - HIGHLVLADMINS-HIGHLVL-ADMINS-WLMDB
Service Subclass: Statistics: Coordinator ac...
Show statistical information
Units
120000
100000
80000
60000
40000
20000
9/1/07 6:00 PM
10/1/07 11:00 AM
time
Day
Week
Month
3 Months
Year

Show Data for : GLOBAL

9/23/07 4:46:20 PM

9/11/07 10:25:31 AM — 10/1/07 12:26:51 PM

0:00:20

10/1/07 10:56:50 AM

Counter path: Cost estimate (timerons) - HIGHLVLADMINS-HIGHLVL-ADMINS-WLMDB

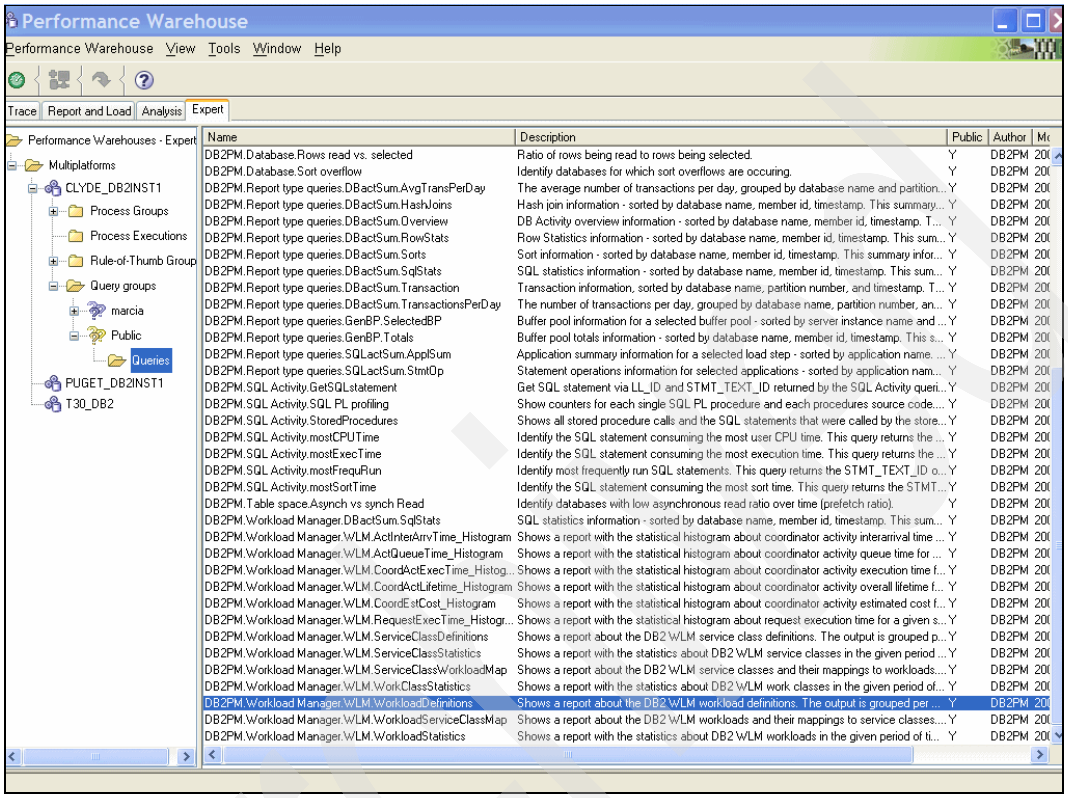
Data collected in Performance Warehouse database from Oct 1, 2006 12:00:00 PM to Oct 1, 2007 11:00:00 AM

Figure 9-28 PWH Trend Analysis chart - timerons

Viewing WLM statistics with PE Performance Warehouse

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 9-29, 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.



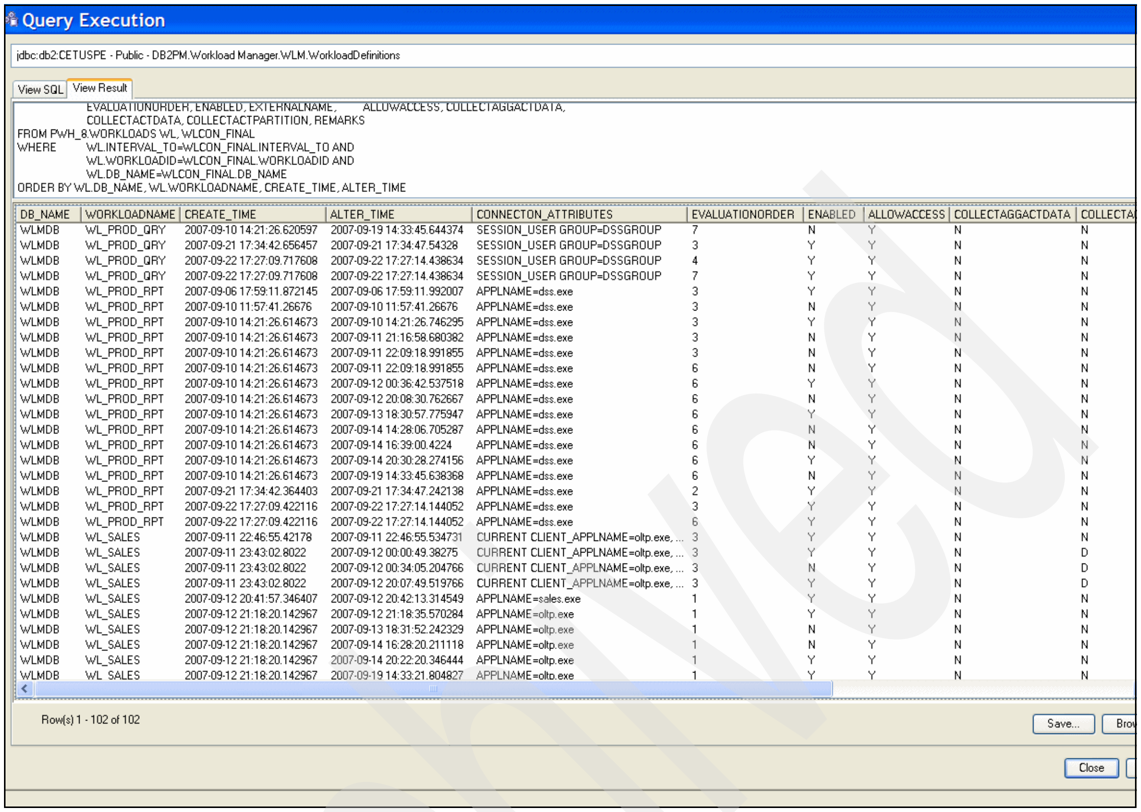
Name	Description	Public	Author	Mo
DB2PM.Database.Rows read vs. selected	Ratio of rows being read to rows being selected.	Y	DB2PM	20
DB2PM.Database.Sort overflow	Identify databases for which sort overflows are occuring.	Y	DB2PM	20
DB2PM.Report type queries.DBactSum.AvgTransPerDay	The average number of transactions per day, grouped by database name and partition...	Y	DB2PM	20
DB2PM.Report type queries.DBactSum.HashJoins	Hash join information - sorted by database name, member id, timestamp. This summary...	Y	DB2PM	20
DB2PM.Report type queries.DBactSum.Overview	DB Activity overview information - sorted by database name, member id, timestamp. T...	Y	DB2PM	20
DB2PM.Report type queries.DBactSum.RowStats	Row Statistics information - sorted by database name, member id, timestamp. This sum...	Y	DB2PM	20
DB2PM.Report type queries.DBactSum.Sorts	Sort information - sorted by database name, member id, timestamp. This summary infor...	Y	DB2PM	20
DB2PM.Report type queries.DBactSum.SqlStats	SQL statistics information - sorted by database name, member id, timestamp. This sum...	Y	DB2PM	20
DB2PM.Report type queries.DBactSum.Transaction	Transaction information, sorted by database name, partition number, and timestamp. T...	Y	DB2PM	20
DB2PM.Report type queries.DBactSum.TransactionsPerDay	The number of transactions per day, grouped by database name, partition number, an...	Y	DB2PM	20
DB2PM.Report type queries.GenBP.SelectedBP	Buffer pool information for a selected buffer pool - sorted by server instance name and ...	Y	DB2PM	20
DB2PM.Report type queries.GenBP.Totals	Buffer pool totals information - sorted by database name, member id, timestamp. This s...	Y	DB2PM	20
DB2PM.Report type queries.SQLactSum.ApplSum	Application summary information for a selected load step - sorted by application name. ...	Y	DB2PM	20
DB2PM.Report type queries.SQLactSum.StmtOp	Statement operations information for selected applications - sorted by application nam...	Y	DB2PM	20
DB2PM.SQL Activity.GetSQLstatement	Get SQL statement via LL_ID and STMT_TEXT_ID returned by the SQL Activity queri...	Y	DB2PM	20
DB2PM.SQL Activity.SQL PL profiling	Show counters for each single SQL PL procedure and each procedures source code....	Y	DB2PM	20
DB2PM.SQL Activity.StoredProcedures	Shows all stored procedure calls and the SQL statements that were called by the store...	Y	DB2PM	20
DB2PM.SQL Activity.mostCPUTime	Identify the SQL statement consuming the most user CPU time. This query returns the ...	Y	DB2PM	20
DB2PM.SQL Activity.mostExecTime	Identify the SQL statement consuming the most execution time. This query returns the ...	Y	DB2PM	20
DB2PM.SQL Activity.mostFrequRun	Identify most frequently run SQL statements. This query returns the STMT_TEXT_ID o...	Y	DB2PM	20
DB2PM.SQL Activity.mostSortTime	Identify the SQL statement consuming the most sort time. This query returns the STMT...	Y	DB2PM	20
DB2PM.Table space.Asynch vs synch Read	Identify databases with low asynchronous read ratio over time (prefetch ratio).	Y	DB2PM	20
DB2PM.Workload Manager.DBactSum.SqlStats	SQL statistics information - sorted by database name, member id, timestamp. This sum...	Y	DB2PM	20
DB2PM.Workload Manager.WLM.ActInterArrvTime_Histogram	Shows a report with the statistical histogram about coordinator activity interarrival time ...	Y	DB2PM	20
DB2PM.Workload Manager.WLM.ActQueueTime_Histogram	Shows a report with the statistical histogram about coordinator activity queue time for ...	Y	DB2PM	20
DB2PM.Workload Manager.WLM.CoordActExecTime_Histog...	Shows a report with the statistical histogram about coordinator activity execution time f...	Y	DB2PM	20
DB2PM.Workload Manager.WLM.CoordActLifetime_Histogram	Shows a report with the statistical histogram about coordinator activity overall lifetime f...	Y	DB2PM	20
DB2PM.Workload Manager.WLM.CoordEstCost_Histogram	Shows a report with the statistical histogram about coordinator activity estimated cost f...	Y	DB2PM	20
DB2PM.Workload Manager.WLM.RequestExecTime_Histogr...	Shows a report with the statistical histogram about request execution time for a given s...	Y	DB2PM	20
DB2PM.Workload Manager.WLM.ServiceClassDefinitions	Shows a report about the DB2 WLM service class definitions. The output is grouped p...	Y	DB2PM	20
DB2PM.Workload Manager.WLM.ServiceClassStatistics	Shows a report with the statistics about DB2 WLM service classes in the given period ...	Y	DB2PM	20
DB2PM.Workload Manager.WLM.ServiceClassWorkloadMap	Shows a report about the DB2 WLM service classes and their mappings to workloads....	Y	DB2PM	20
DB2PM.Workload Manager.WLM.WorkClassStatistics	Shows a report with the statistics about DB2 WLM work classes in the given period of...	Y	DB2PM	20
DB2PM.Workload Manager.WLM.WorkloadDefinitions	Shows a report about the DB2 WLM workload definitions. The output is grouped per ...	Y	DB2PM	20
DB2PM.Workload Manager.WLM.WorkloadServiceClassMap	Shows a report about the DB2 WLM workloads and their mappings to service classes....	Y	DB2PM	20
DB2PM.Workload Manager.WLM.WorkloadStatistics	Shows a report with the statistics about DB2 WLM workloads in the given period of ti...	Y	DB2PM	20

Figure 9-29 PE Performance Warehouse - Predefined Queries

To execute a query as-is, right-click the query and select **EXECUTE** from the context menu. In many cases, you might want to modify the query slightly to suit your own needs, but we do not explore that in this section.

> **Note:** To see more examples showing how to use the DB2 Performance Expert Performance Warehouse queries and reports, refer to the 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 9-30. 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,
              COLLECTACTDATA, COLLECTACTPARTITION, REMARKS
FROM PWH_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	CONNECTON_ATTRIBUTES	EVALUATIONORDER	ENABLED	ALLOWACCESS	COLLECTAGGACTDATA	COLLECTA
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=dss.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=dss.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=dss.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=dss.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=dss.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=dss.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=dss.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=dss.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=dss.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=dss.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=dss.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=dss.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=dss.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=dss.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=dss.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=dss.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_APPLNAME=oltp.exe, ...	3	Y	Y	N	N
WLMDB	WL_SALES	2007-09-11 23:43:02.8022	2007-09-12 00:00:49.38275	CURRENT CLIENT_APPLNAME=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_APPLNAME=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_APPLNAME=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=sales.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

Figure 9-30 Performance Warehouse query results - Workload Definition

9.4 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.

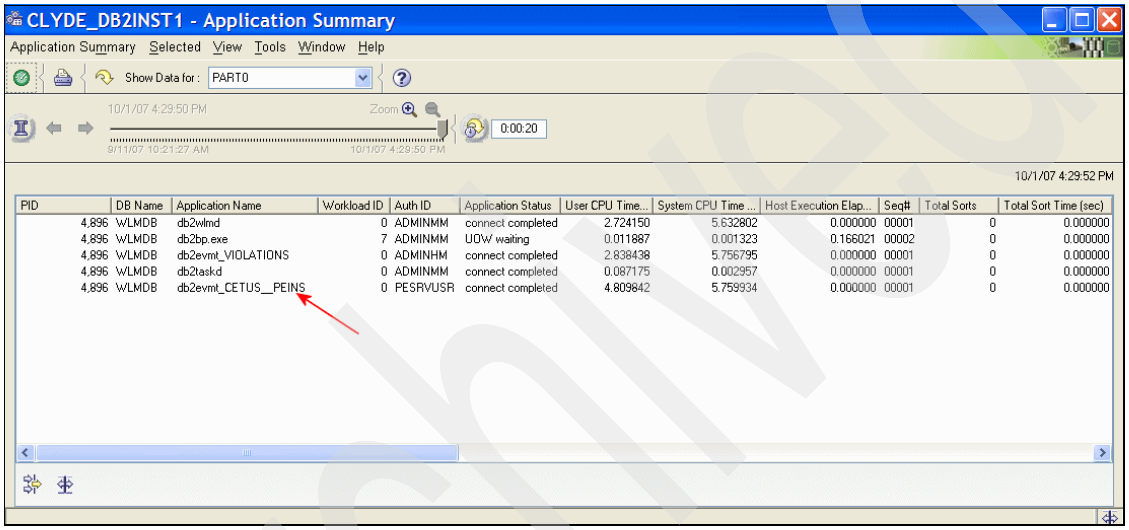
Database Configuration parameter WLM_COLLECT_INT

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, you should adjust this parameter on the catalog partition.

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 9-31. In this case the PE Server hostname is CETUS and the PE instance name is PEINST, so the event monitor name is CETUS__PEINST (although the full name is truncated in the snapshot output).



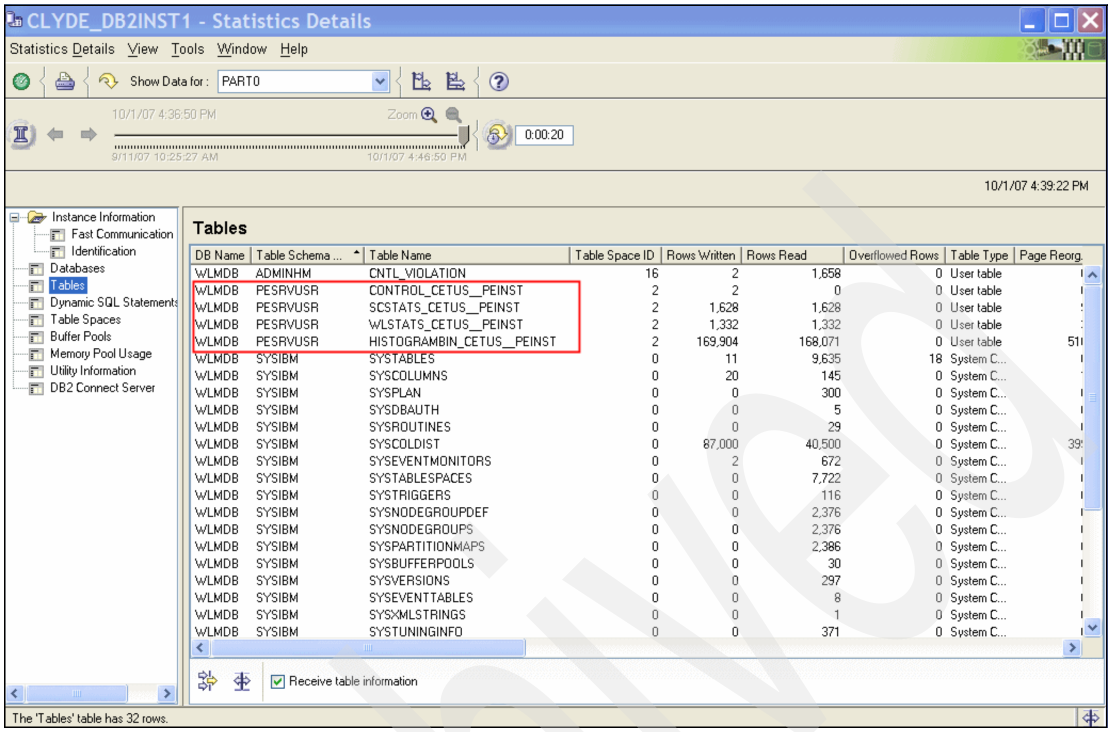
PID	DB Name	Application Name	Workload ID	Auth ID	Application Status	User CPU Time...	System CPU Time ...	Host Execution Elap...	Seq#	Total Sorts	Total Sort Time (sec)
4,896	WLMDB	db2wlmd	0	ADMINMM	connect completed	2.724150	5.632802	0.000000	00001	0	0.000000
4,896	WLMDB	db2bp.exe	7	ADMINMM	UOW waiting	0.011887	0.001323	0.166021	00002	0	0.000000
4,896	WLMDB	db2evmt_VIOLATIONS	0	ADMINHM	connect completed	2.838438	5.756795	0.000000	00001	0	0.000000
4,896	WLMDB	db2taskd	0	ADMINMM	connect completed	0.087175	0.002957	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 9-31 PE Application Summary showing active event monitor

Notice that there is another event monitor active on this instance that is 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 Threshold Violations type. The PE event monitor can coexist with activity and threshold violation 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 that 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; see Figure 9-32 for an example.



Tables

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	51
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	SYSEVENTMONITORS	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	SYSEVENTTABLES	0	0	8	0	System C...	
WLMDB	SYSIBM	SYSXMLSTRINGS	0	0	1	0	System C...	
WLMDB	SYSIBM	SYSTUNINGINFO	0	0	371	0	System C...	

The 'Tables' table has 32 rows.

Figure 9-32 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 that 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 9-33. In this example we look at the table BASIC_MON_HISTOGRAMS, which was used for examples in earlier sections of this book (when not discussing PE).

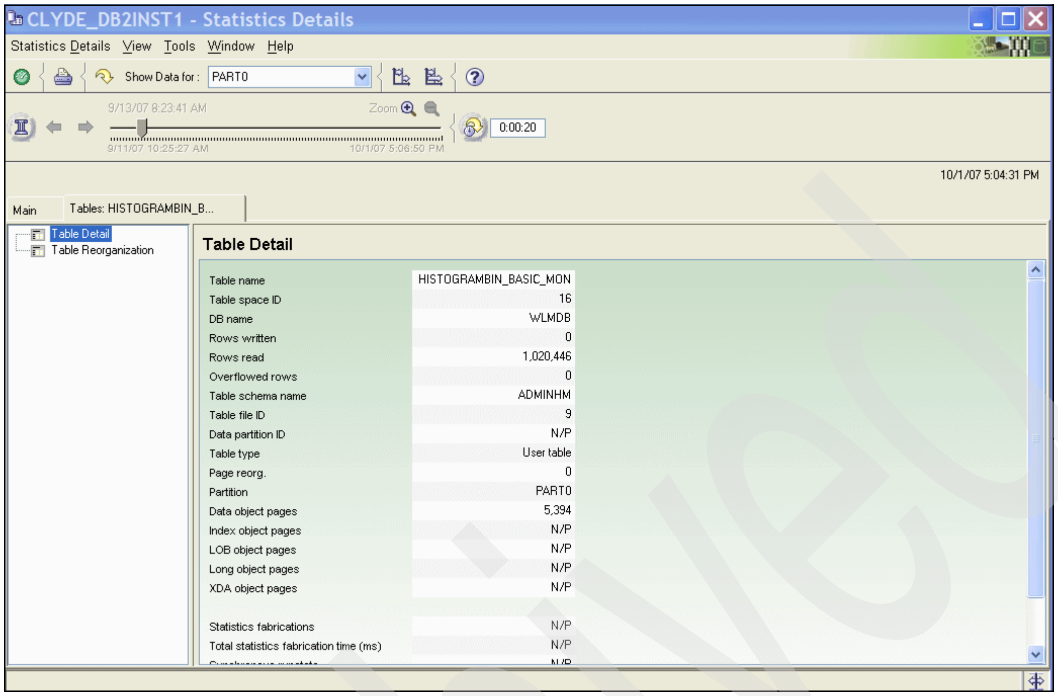
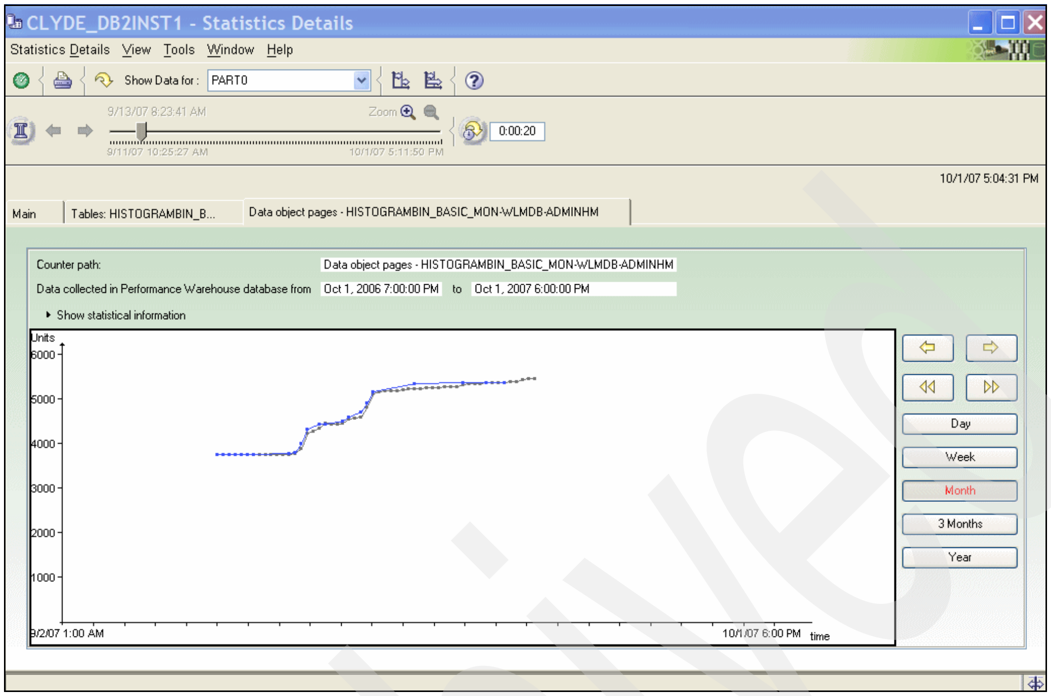


Table Detail

Table name	HISTOGRAMBIN_BASIC_MON
Table space ID	16
DB name	WLMDB
Rows written	0
Rows read	1,020,446
Overflowed rows	0
Table schema name	ADMINHM
Table file ID	9
Data partition ID	N/P
Table type	User table
Page reorg.	0
Partition	PART0
Data object pages	5,394
Index object pages	N/P
LOB object pages	N/P
Long object pages	N/P
XDA object pages	N/P
Statistics fabrications	N/P
Total statistics fabrication time (ms)	N/P

Figure 9-33 PE Table detail for non-PE histogram event monitor table

In our case, we intentionally did not clear out this table because we were conducting tests, but if you look at Figure 9-34, 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.

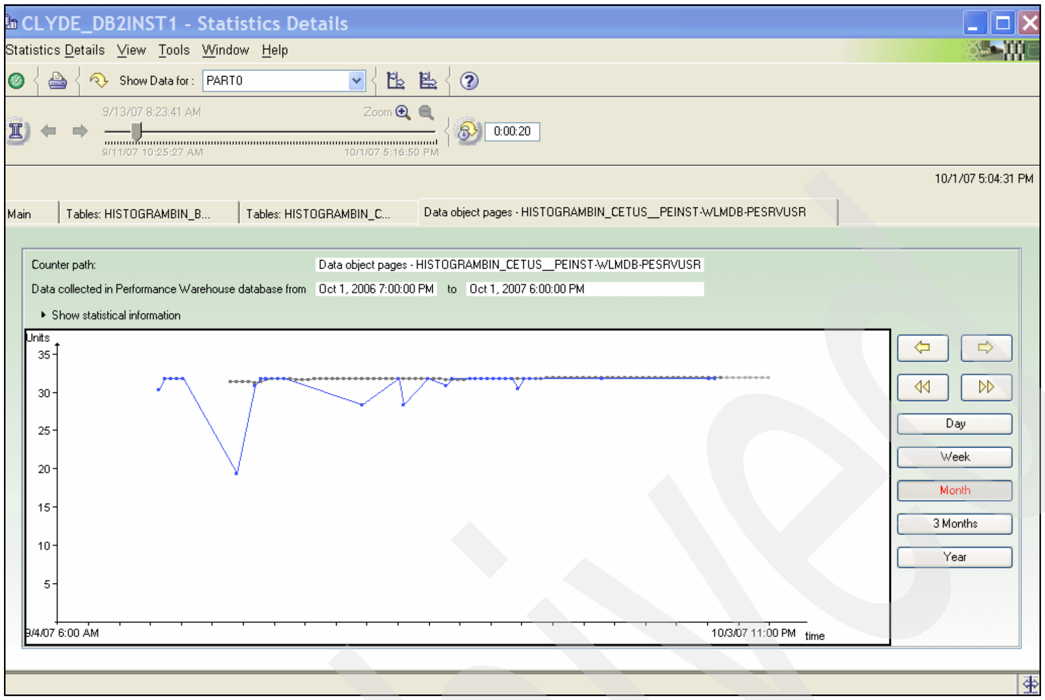


Counter path: Data object pages - HISTOGRAMBIN_BASIC_MON-WLMDB-ADMINHM

Data collected in Performance Warehouse database from Oct 1, 2006 7:00:00 PM to Oct 1, 2007 6:00:00 PM

Figure 9-34 Trend chart of manual event monitor table size

Compare this to the PE event monitor table size, as shown in Figure 9-35, where you can see the table is much less volatile in its size.


CLYDE_DB2INST1 - Statistics Details
Statistics Details View Tools Window Help
Show Data for: PART0
9/13/07 8:23:41 AM
Zoom
9/11/07 10:25:27 AM
10/1/07 5:16:50 PM
0:00:20
10/1/07 5:04:31 PM
Main
Tables: HISTOGRAMBIN_B...
Tables: HISTOGRAMBIN_C...
Data object pages - HISTOGRAMBIN_CETUS__PEINST-WLMDB-PESRVUSR
Show statistical information
Units
35
30
25
20
15
10
5
9/4/07 6:00 AM
10/3/07 11:00 PM
time
Day
Week
Month
3 Months
Year

Counter path: Data object pages - HISTOGRAMBIN_CETUS__PEINST-WLMDB-PESRVUSR

Data collected in Performance Warehouse database from Oct 1, 2006 7:00:00 PM to Oct 1, 2007 6:00:00 PM

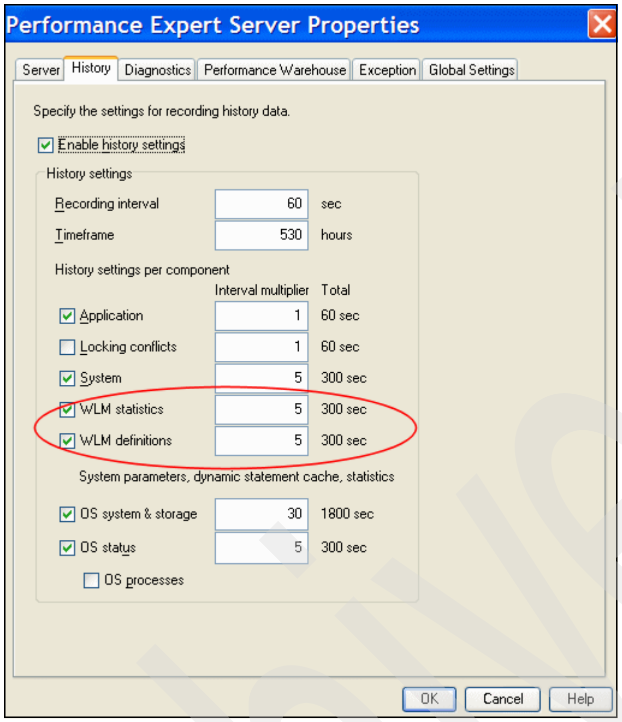
Figure 9-35 Trend chart of PE event monitor table size

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 earlier 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 9-36.

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.



Performance Expert Server Properties

Server | History | Diagnostics | Performance Warehouse | Exception | Global Settings

Specify the settings for recording history data.

☑ Enable history settings

History settings

Recording interval	60	sec
Timeframe	530	hours

History settings per component

	Interval multiplier	Total
☑ Application	1	60 sec
☐ Locking conflicts	1	60 sec
☑ System	5	300 sec
☑ WLM statistics	5	300 sec
☑ WLM definitions	5	300 sec

System parameters, dynamic statement cache, statistics

☑ OS system & storage	30	1800 sec
☑ OS status	5	300 sec
☐ OS processes		

OK | Cancel | Help

Figure 9-36   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.

Modifying WLM monitoring settings

In Fixpack1, PE introduces the ability to modify the monitoring settings on an existing WLM object, such as Service Class. You can do this manually using, for example, the ALTER SERVICE CLASS statement, but now you can also control this from within the PE GUI. You can change the level of data collection with the COLLECT AGGREGATE clause, either ACTIVITY DATA or REQUEST DATA.

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, a snapshot is not used, so when you are using the GUI screens you are only 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 it could be older data. You can always see the date and time of the data on the screen by looking at the upper right portion of the screen. (This is true for all PE screens, by the way, and 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, then you will get more comfortable with PE.

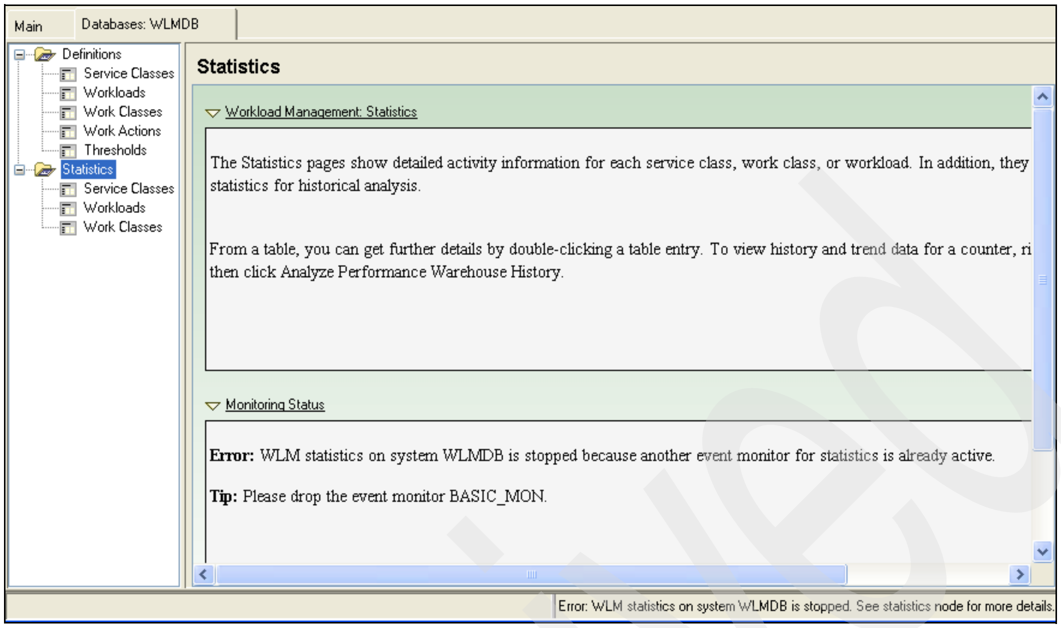
Coexisting event monitors

As described in earlier chapters of this book, you can create your own event monitors for capturing WLM data. There are three types of WLM event monitors: statistics, activities, and threshold violations. PE currently only uses the statistics event monitor, so there are no coexistence issues with creating your own activity or threshold event monitor.

Note: In PE V3.1 Fixpack1, you can now also create and activate Activity event monitors from PE. You run these event monitors on demand, and for a fixed duration. The Activity event monitor can be expensive, so you should not run more than one at a time, but there is no coexistence issue with the Statistics event monitor, or with other event monitors that may be defined but inactive.

If you create and activate your own statistics event monitor, which also assumes you will modify the WLM_COLLECT_INT parameter, then PE will not collect the WLM data. Your manual event monitor will supersede the PE event monitor.

However, it is best to avoid this situation altogether and simply allow 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 9-37, and tell you to drop it. Although there is no real impact in keeping them both defined, your results may be unpredictable if you try to activate both of them.



Statistics

Workload Management: Statistics

The Statistics pages show detailed activity information for each service class, work class, or workload. In addition, they statistics for historical analysis.

From a table, you can get further details by double-clicking a table entry. To view history and trend data for a counter, ri then click Analyze Performance Warehouse History.

Monitoring Status

Error: WLM statistics on system WLMDB is stopped because another event monitor for statistics is already active.

Tip: Please drop the event monitor BASIC_MON.

Error: WLM statistics on system WLMDB is stopped. See statistics node for more details.

Figure 9-37 PE warning for duplicate event monitor

Administering a DB2 workload management environment

This chapter discusses the key points to check in administering a DB2 workload management environment. As you gain familiarity with how DB2 Workload Manager (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. After you migrate all of your WLM settings to full production, you need to periodically purge old data, back up the WLM settings, and take note of the tools needed in problem diagnosis.

We discuss the following topics in this chapter:

- WLM logs and maintenance
- WLM problem diagnosis
- WLM backup and recovery
- WLM authorization

10.1 Workload Manager 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 and files. To see the WLM-specific messages being written out, check the DB2 diagnostic log, db2diag.log, and the administration notification log, <DB2 instance name>.nfy.

DB2 activities or statistics event monitors will be deactivated if there is no space for DB2 to write the event monitor information to. 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.

10.2 Workload Manager problem diagnosis

When a problem is encountered in WLM, the symptoms of the problem must first be examined. You can ask the following questions 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 result if 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_OCCURRENCE_ACTIVITIES table function.

  – Get the workload information in memory by using the **`db2pd -db <database name> -workloads`** command.

- Is the problem related to a WLM service class using more than a fair share of agents?

  One possible cause of a problem is that 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 10-1 illustrates the use of the WLM_GET_SERVICE_CLASS_AGENTS table function to determine how many agents are working for each service class.

Example 10-1 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 (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 whether 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 10-2 can be used to get a high level view 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 10-2 Query to get a high level view 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 WLM_GET_SERVICE_CLASS_AGENTS, WLM _GET_SERVICE_CLASS_STATS, and the db2pd utility, to obtain the necessary information about 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 you obtain 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
- Service classes including superclasses and the subclasses
- 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
- Threshold information

Example 10-3 shows how to invoke the **db2pd** options for WLM.

Example 10-3 Using the db2pd utility to get WLM information

```
db2pd -db WLMDB -workloads -workactionsets -workclasssets
```

Example 10-4 is a sample portion of the output of **db2pd**.

Example 10-4 Sample output from db2pd

```
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.
```

10.3 Workload Manager 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 after the DB2 catalog table space is backed up.

You also can use the **db2look** utility to save the WLM definitions to a file as a backup or as system documentation. Save the WLM definition in a file allows you to restore the WLM definitions to previous version without restoring the entire database. The following command saves the WLM object definitions in a file:

```
db2look -d WLMDB -e -wlm -xd -o wlm.definitions.out
```

This saves all the CREATE and ALTER statements for WLM objects, including all WLM CREATE event monitor statement, as well as GRANT USAGE ON WORKLOAD statements.

10.4 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 as well. They may result in greater or less amount of that resource. 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, 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 that session user, group, or role 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 an 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 10-5 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 10-5 Examples of the GRANT USAGE statement

```
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 10-6 shows how the USAGE privileges granted in Example 10-5 can be revoked.

Example 10-6 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;
```

11

Query Patroller and DB2 Governor

Prior to DB2 9.5, DB2 workload management consisted of Query Patroller (QP) and the DB2 Governor. With the new offering of the DB2 Workload Manager, you may be wondering why there was a need to build a separate offering. And 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 in this chapter:

- A brief description of QP and Governor features
- A comparison of differences between workload management solutions of QP and Governor, and what is available in DB2 9.5
- How DB2 WLM, QP, and the Governor can coexist on the same data server
- How to approach the task of migrating from QP and the Governor to the new WLM features

11.1 Query Patroller and DB2 Governor background

In this section, we provide background information about Query Patroller and the DB2 Governor. We also explain the differences between DB2 WLM, Query Patroller, and DB2 Governor.

11.1.1 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 (based on the estimated cost for the query, or timeron value). 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 11-1 on page 323 illustrates the Query Patroller environment. As queries enter the data server, some will be managed by QP, and QP will decide when to flow them into the system. Other queries (usually short-running queries) will bypass QP to avoid excess overhead, and those will flow directly into the system.

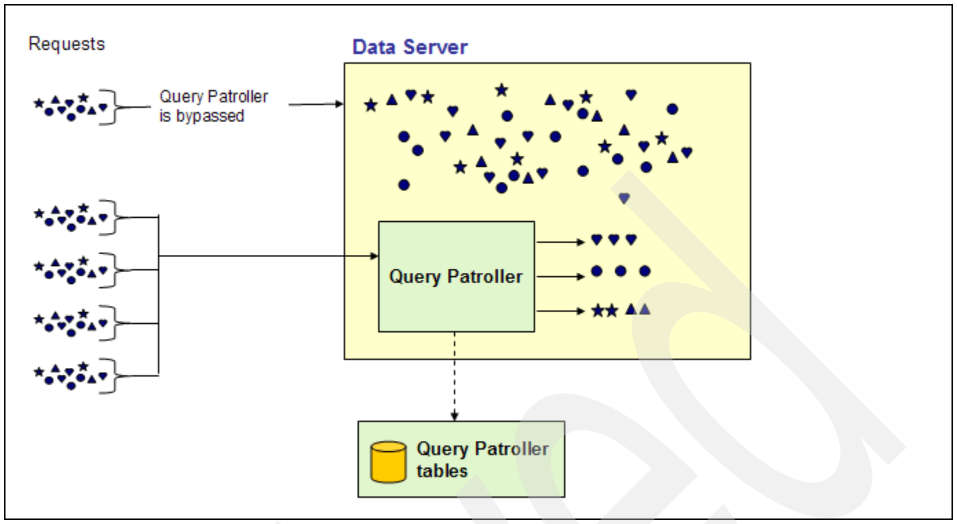


Figure 11-1 Query Patroller environment

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 (such as really short queries or queries 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 too large to run at all, and needs to be looked at and possibly scheduled to run at a more appropriate time).
- After 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).

11.1.2 DB2 Governor

Query Patroller's predictive query management approach is fundamentally based on estimates, and no matter how accurate those estimates may be, they are still estimates—which means they could be wrong. For example, 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).

This is where the DB2 Governor nicely complements Query Patroller with its reactive approach to workload management. After QP lets the query loose to execute, the DB2 Governor then watches for certain thresholds during the execution that can result in 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

11.1.3 Differences between QP, Governor, and WLM

Although 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 11-1 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 11-1 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.

Query Patroller/DB2 Governor	DB2 Workload Manager
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 (for example, DML, DDL, Load, and so on). The level of workload management is customized for each service class.
After 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.
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 activity, a less harsh option is available to simply capture details on the activity for future analysis.
QP and the Governor sit on top of the DB2 data server engine. QP has to communicate with a QP controller process to make management decisions.	Integrated into the DB2 data server engine itself, thereby minimizing management overhead.

11.2 Coexistence

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 effort. 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.

Note: To dispel any misconceptions, keep in mind that although QP and the Governor may not be the future strategic direction for workload management in DB2, 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 does 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 11-2 illustrates the Query Patroller in a default WLM environment.

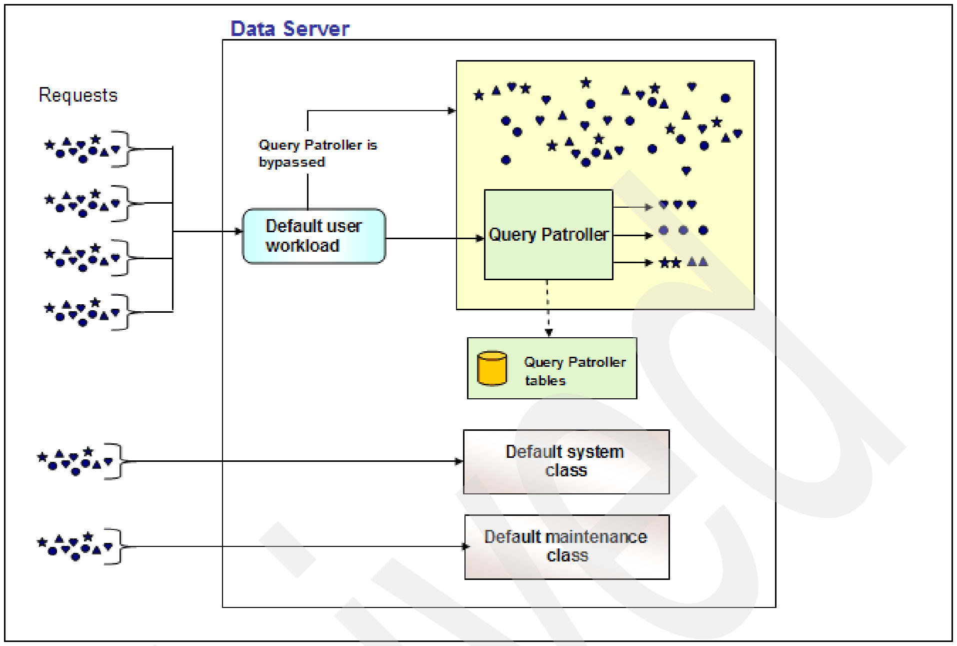


Figure 11-2 Query Patroller in a default WLM environment

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, because they will bypass those tools completely.

11.3 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 how to approach the transition of managing workloads with Query Patroller and the Governor to managing workloads with DB2 Workload Manager.

11.3.1 No migration tool

Many Query Patroller customers have achieved success with managing workloads using query classes and other submitter thresholds. A significant amount of time may have been invested in building the configuration, and these customers are comfortable with what they have built. It is only natural that they 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 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 straightforward one would do a better job.

11.3.2 Reexamining 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 historical analysis of database object usage (what tables are accessed, for example)
- Capture database activity information for charge back accounting
- Simply maximize the throughput of data requests on the system

11.3.3 Migrating from a QP environment - considerations

As previously mentioned, to be well-understood and efficiently tuned, effective workload management is based on the following four phases:

- 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 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 a WLM configuration.

Identification

Query Patroller identifies and classifies database activities in two ways. The first way is based on a range of estimated query cost (for example, group work by small-, medium-, or large-sized queries). The second way 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 this 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 value, however, can be quite difficult to determine. How do you really know when you start to encounter resource contention? For example, is it when a user submits more than five queries? Does it depend on the time of day? Does it depend on the 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. This generalization has to be lenient enough to avoid imposing 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 involved in arriving at 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: Do not use concurrency thresholds as the first option for workload management. Instead, look at controlling the CPU and prefetch priority options on service classes. If you are on the AIX platform, consider using AIX WLM for a more granular control of CPU.

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 the estimated cost of the query.

There are often three groups: very short transaction queries; very large report generation queries; and 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 large, resource-intensive queries from consuming a disproportionate amount of resources, which 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 also exist for query classes. However, not only do you have to determine a suitable maximum concurrency, but also you 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 from users 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.

This is a prime example of stepping back and reexamining 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, however, these different types of queries rarely come from the same application.

For example, short queries could be coming from an application executing mass transactions, OLTP, and even ETL activity. Long-running queries might be heavy report generation applications. So, why not create a workload for the applications executing the short transactions, and another workload for the applications driving the heavy reports, and then map the work to separate service classes?

In this way you can control the resources for each service class where you can keep the large queries throttled back by limiting CPU or I/O, or both. Concurrency thresholds are still available on a service class, so you can continue to limit the number of activities for an application to 5.

Migration action: Initially, do not try to implement a query class-like environment in DB2 WLM. Instead, 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 managing the setting in QP to exclude the set of short queries from being managed. This cost is required because of the potentially huge volume of short queries all communicating with the query controller, and because all those queries are 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 layer), 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 queries that are estimated to be so large that you do not want to even start executing the SQL. A typical example would be when a user mistakenly issues a query containing a Cartesian join: QP puts such a query in a held state, does not execute it, and returns an error to the submitting application.

It is very useful 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, as shown:

```
CREATE WORK CLASS SET "queries"
   (WORK CLASS "very big query"
      WORK TYPE DML
      FOR TIMERONCOST FROM 10000000 TO UNBOUNDED)
```

A few actions can be taken against these very large 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 such cases, stopping the query may be too radical an action. Collecting activity details in the event monitor might be a better solution, so that you can analyze the potential problem later. To do this, create the corresponding work action set as shown:

```
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)
```

However, if you really do want to stop those very large queries from ever running, you could create a work action set something like the one shown here:

```
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: Do not be concerned about trying to implement a minimum cost to manage setting in WLM. Identifying large queries is still very useful, but consider the options of simply collecting detailed information about the very large queries, or limiting the scope of what you collect to a specific service class 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 (for example, batch ETL scripts).

Queries and applications 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 DB2_QP_BYPASS_APPLICATIONS, DB2_QP_BYPASS_USERS, or DB2_QP_BYPASS_COST 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). It also does not (by default) write individual records of activity details to a table. So, WLM does not have the overhead issues of QP, and no option to bypass WLM is provided. (It is also a fundamental architectural point to have all database activities run in a service class.)

Migration action: There is no migration action. There is no need for a mechanism to allow database activities to bypass WLM.

Managing with connection pooling

Connection pooling is a very common practice found in many vendor products that access data from DB2. Basically, the vendor 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 permitted to obtain 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, suppose user JOE is using a vendor application to build a report. JOE connects to reportapp.exe with a user ID 'JOE'. The vendor application determines JOE is permitted 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, that is, not without also controlling every other user who is using the connection from the connection pool.

In contrast, DB2 WLM has the client attributes available for proper identification and management of queries. The application at the middle tier could either make a sqleseti API call or wlm_set_client_info stored procedure call to set one of the client attributes before it issues the SQL.

In this example, even though the connection is established using the 'VENDORID' session ID, the vendor application can call the sqleseti 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 this user's resource consumption, or by setting thresholds on the requests.

After 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 sqleseti API to set client attributes to handle connection pooling in a 3-tier application environment.

11.3.4 Migrating from a DB2 Governor environment - considerations

The DB2 Governor has its configuration stored in a control file. There are a few parameters in particular that are 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 several resource events to look at as well.

The rowssel event is used to indicate that after a certain number of data rows are returned to the application, then 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 that the Governor is 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 allowing the threshold continue to execute, and then using 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 map 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.

11.3.5 Historical information in QP control tables

One advantage existing Query Patroller customers have over customers new to WLM is the fact that they already have a (potentially very large) set of query activity. This is a useful 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 11-2 displays these columns, along with suggestions about what to consider when identifying database activities through workloads.

Table 11-2 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.
APPLICATION	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).
CLIENT_USER_ID	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.
CLIENT_ACCOUNT_ID	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).

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 (for instance, 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 after those workload occurrences are assigned to a service class, Query Patroller will no longer be aware of those queries.

Figure 11-3 illustrates how the workload management environment would look after you start isolating activities based on the QP history.

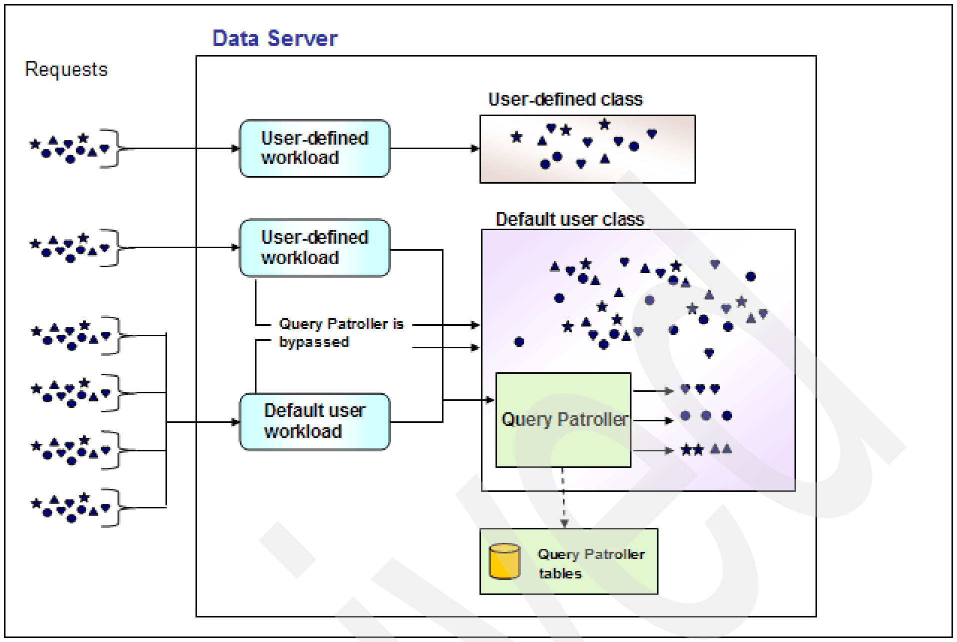


Figure 11-3 Establishing user-defined service classes with QP data

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 11-3.

Table 11-3 TRACK_QUERY_INFO columns that map to other WLM functions

TRACK_QUERY_INFO column	**WLM mgration 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 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 mgration 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 in determining the overall distribution of the query workload. Analysis can be performed on that distribution to isolate 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 (for example, 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; 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 IBM Redbooks” on page 344. Note that some of the documents referenced here may be available in softcopy only.

- *AIX 5L Workload Manager (WLM)*, SG24-5977
- *DB2 Performance Expert for Multiplatforms V2.2*, SG24-6470
- *Leveraging DB2 Data Warehouse Edition for Business Intelligence*, SG24-7274

Other publications

These publications are also relevant as further information sources:

IBM - DB2 9.5

- *What's New*, SC23-5869
- *Administrative API Reference,* SC23-5842
- *Administrative Routines and Views,* SC23-5843
- *Call Level Interface Guide and Reference, Volume 1,* SC23-5844
- *Call Level Interface Guide and Reference, Volume 2,* SC23-5845
- *Command Reference*, SC23-5846
- *Data Movement Utilities Guide and Reference*, SC23-5847
- *Data Recovery and High Availability Guide and Reference*, SC23-5848
- *Data Servers, Databases, and Database Obj*ects Guide, SC23-5849
- *Database Security Guide*, SC23-5850
- *Developing ADO.NET and OLE DB Applications*, SC23-5851
- *Developing Embedded SQL Applications*, SC23-5852

- *Developing Java Applications*, SC23-5853
- *Developing Perl and PHP Applications*, SC23-5854
- *Developing User-Defined Routines (SQL and External)*, SC23-5855
- *Getting Started with Database Application Development*, GC23-5856
- *Getting Started with DB2 Installation and Administration on Linux and Windows*, GC23-5857
- *Internationalization Guide*, SC23-5858
- *Message Reference, Volume 1*, GI11-7855
- *Message Reference, Volume 2*, GI11-7856
- *Migration Guide*, GC23-5859
- *Net Search Extender Administration and User's Guide*, SC23-8509
- *Partitioning and Clustering Guide*, SC23-5860
- *Query Patroller Administration and User's Guide*, SC23-8507
- *Quick Beginnings for IBM Data Server Clients*, GC23-5863
- *Quick Beginnings for DB2 Servers*, GC23-5864
- *Spatial Extender and Geodetic Data Management Feature User's Guide and Reference*, SC23-8508
- *SQL Reference, Volume 1*, SC23-5861
- *SQL Reference, Volume 2*, SC23-5862
- *System Monitor Guide and Reference*, SC23-5865
- *Troubleshooting Guide*, GI11-7857
- *Tuning Database Performance*, SC23-5867
- *Visual Explain Tutorial*, SC23-5868
- *Workload Manager Guide and Reference*, SC23-5870
- *pureXML Guide*, SC23-5871
- *XQuery Reference*, SC23-5872
- *DB2 Performance Expert for Mulitplatforms Installation and Configuration Guide*, SC19-1174

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/db2luw/v9r5/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 IBM Redbooks

You can search for, view, or download IBM 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

IBM

Redbooks

DB2 Workload Manager for Linux, UNIX, and Windows

(0.5" spine)
0.475"<->0.875"
250 <-> 459 pages

IBM®

DB2 Workload Manager for Linux, UNIX, and Windows

Redbooks®

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 higher volumes with minimal overhead. It is also enabled for tighter integration with external workload management products, such as those provided by AIX WLM. This IBM Redbooks publication discusses the features and functions of DB2 Workload Manager for Linux, UNIX, and Windows. It describes DB2 WLM architecture, components, and WLM-specific SQL statements. It demonstrates installation, WLM methodology for customizing the DB2 WLM environment, new workload monitoring table functions, event monitors, and stored procedures. It provides examples and scenarios using DB2 WLM to manage database activities in DSS and OLTP mixed database systems, so you learn about these advanced workload management capabilities and see how they can be used to explicitly allocate CPU priority, detect and prevent “runaway” queries, and closely monitor database activity in many different ways. Using Data Warehouse Edition Design Studio and DB2 Performance Expert with DB2 WLM is covered. Lastly, the primary differences between Workload Manager and Query Patroller are explained, along with how they interact in DB2 9.5.

SG24-7524-00 ISBN 0738485381