Jornada Técnica IV de DB2 LUW

DB2 9.5. WLM

Ana Isabel Rivera (ana_rivera@es.ibm.com)
IBM DB2 Technical Presales

November 25, 2009

Agenda

- **Workload Manager. Reasons and Objectives.**
- **Workload Manager. Stages and Features.**
- **Best Practices and Examples.**
- **Summary**

Agenda

- **Workload Manager. Reasons and Objectives.**
- Workload Manager. Stages and Features.
- Best Practices and Examples.
- Summary

Workload Management in DB2. Reasons.

- Maintain consistent response times (stability and predictability)
- Protect business critical workloads
- Explicit resource control
- Enforce and measure service level agreement objectives
- Protect the data server from poorly written rogue queries
- Monitoring the entire lifecycle of database workload

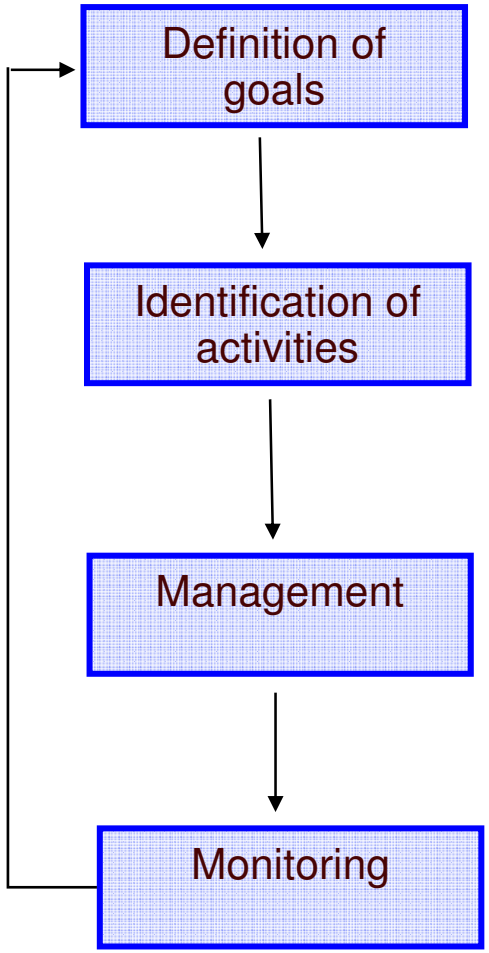
Workload Management in DB2. Objetives

- **The monitoring and control of work executing on a database system to**
  - **Maximize system efficiency/throughput**
  - **Achieve business performance objectives**
- **Prior to DB2 9.5, DB2 provided focused WLM support through a combination of Query Patroller and DB2 Governor**
  - **Both applications running outside of the DB2 server engine**
- **As of DB2 9.5, DB2 provides focused WLM support through the new integrated workload management capabilities in the DB2 server engine**

Agenda

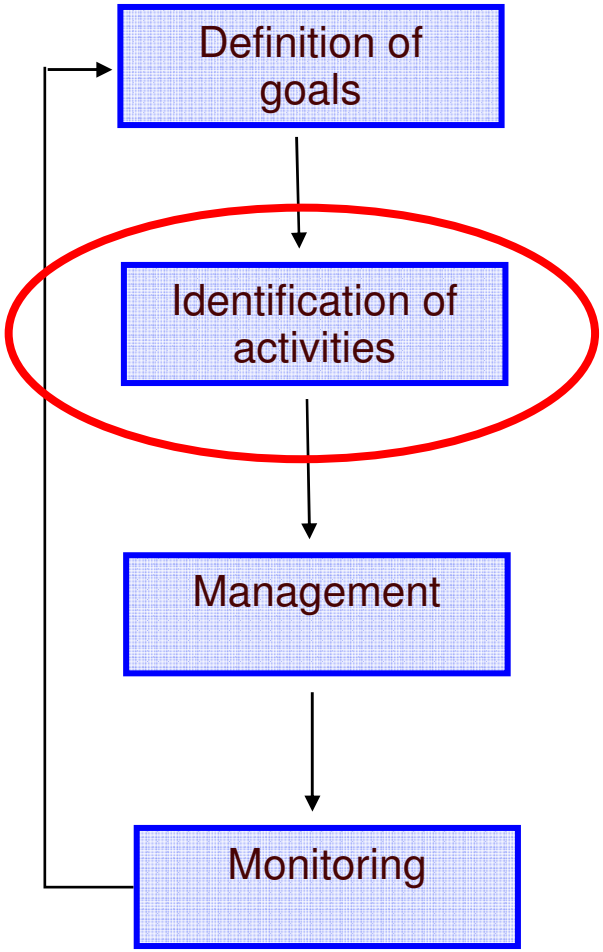
- Workload Manager. Reasons and Objectives.
- **Workload Manager. Stages and Features.**
- Best Practices and Examples.
- Summary

Stages of Workload Management

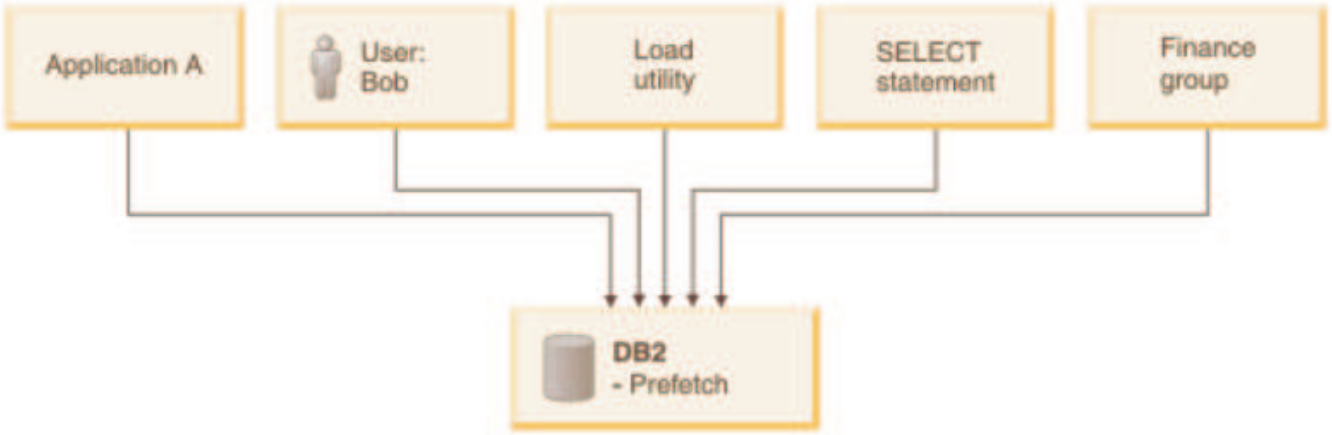


- **Stages of Workload Management**
  - Understanding business goals
  - Identification of work to manage
  - Management
  - Monitoring

Stages. WLM FEATURES: Identification Stage



WLM Features. Identification.



Two new database objects are provided to implement identification:

- WORKLOAD
- WORK CLASS

WLM Features - Workload

- **DB2 Workloads identify incoming work based on its source**
- **Connections are assigned to a workload based on:**
  - Application name
  - System or session authorization ID
  - Group
  - Role
  - Client attributes (user ID, application name, workstation name, or accounting string)
- **Attributes can be set by the application or application server**
  - via C API, Java Methods, SQL stored procedures
- **Work is then directed to a service class**
- **Always a default workload**

WLM Features – Workload Example

CREATE WORKLOAD "Reporting" APPLNAME('Accounts');
GRANT USAGE ON WORKLOAD REPORTING TO PUBLIC;

All connections with application name of 'Accounts'

CREATE WORKLOAD "Summary" SESSION_USER GROUP('Deptmgr')
APPLNAME('Accounts') ;
GRANT USAGE ON WORKLOAD SUMMARY TO PUBLIC;

All connections with application name of 'Accounts' where session user belongs to 'Deptmgr' group

WLM Feature – Work Class/Work Action Set

- **While DB2 Workloads identify work based on its source, work classes identify work based on type. Not mandatory.**
- **Work classes can identify:**
  - Read statements
    - SELECT and XQuery statement consisting of fetches
  - Write statements
    - INSERT, UPDATE, DELETE, and MERGE statements
  - DML (includes Read and Write)
  - DDL
    - Statements that modify or create database objects
  - Load utility
  - CALL
  - Predictive Identification :
    - Estimated cost (range of cost estimates)
    - Estimated cardinality (range of rows returned estimates)

Work Class Set and Work Action Set Syntax

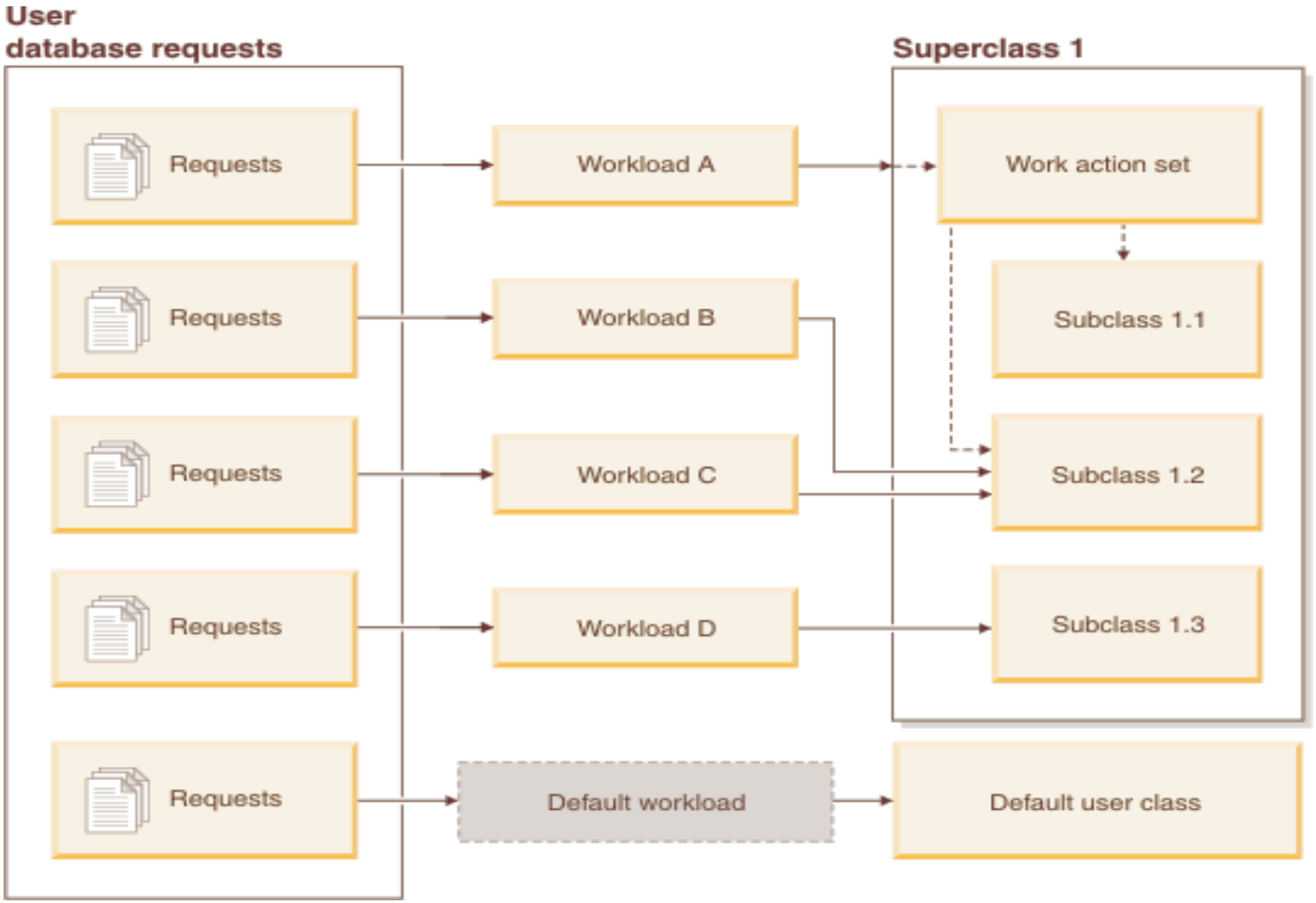
```
CREATE WORK CLASS SET all_class_types
        (WORK CLASS read_wc WORK TYPE READ,
          WORK CLASS write_wc WORK TYPE WRITE,
          WORK CLASS ddl_wc WORK TYPE DDL,
          WORK CLASS call_wc WORK TYPE CALL,
          WORK CLASS load_wc WORK TYPE LOAD,
          WORK CLASS all_wc WORK TYPE ALL POSITION LAST)


CREATE WORK ACTION SET db_was FOR DATABASE
  USING WORK CLASS SET  all_class_types
        (WORK ACTION collect_load_wa ON WORK CLASS load_wc
          COLLECT ACTIVITY DATA WITH DETAILS AND VALUES,
        WORK ACTION collect_ddl_wa ON WORK CLASS ddl_wc
          COLLECT ACTIVITY DATA WITH DETAILS AND VALUES,
        WORK ACTION collect_read_wa ON WORK CLASS read_wc
          COLLECT ACTIVITY DATA WITH DETAILS AND VALUES,
        WORK ACTION stop_large_read_wa on WORK CLASS read_wc
          WHEN ESTIMATEDSQLCOST > 10000 STOP EXECUTION )
```

WLM Feature – Work class/Work action set

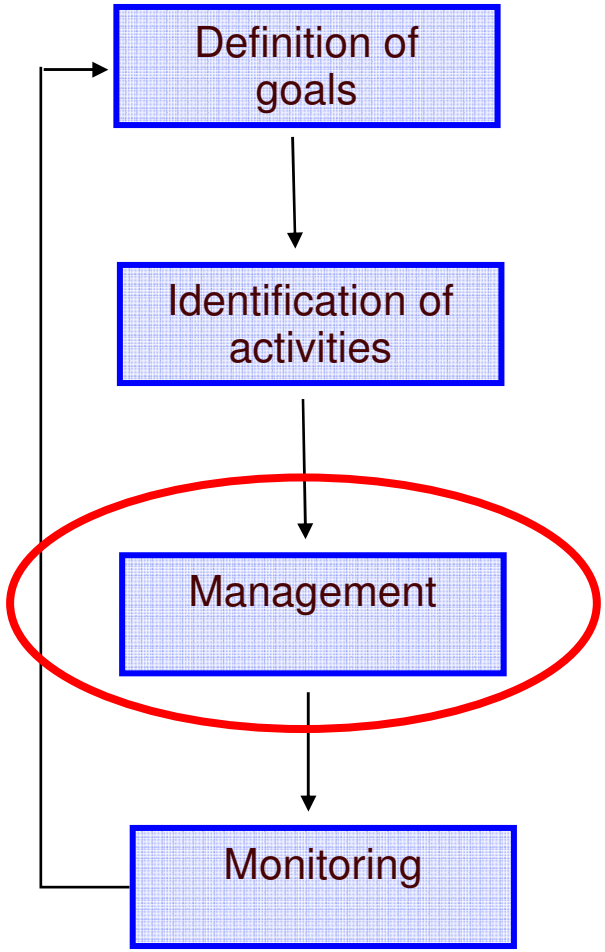
- **Once the type of work is identified, work actions are applied to control the activities.**

- **Possible work actions include:**
  - Map activities to a service subclass (imposes subclass resource management)
  - Prevent execution of activities
  - Collect activity or aggregate data
  - Count activities

WLM Features – Work Action Set Mapping



Stages of Workload Management . WLM FEATURES



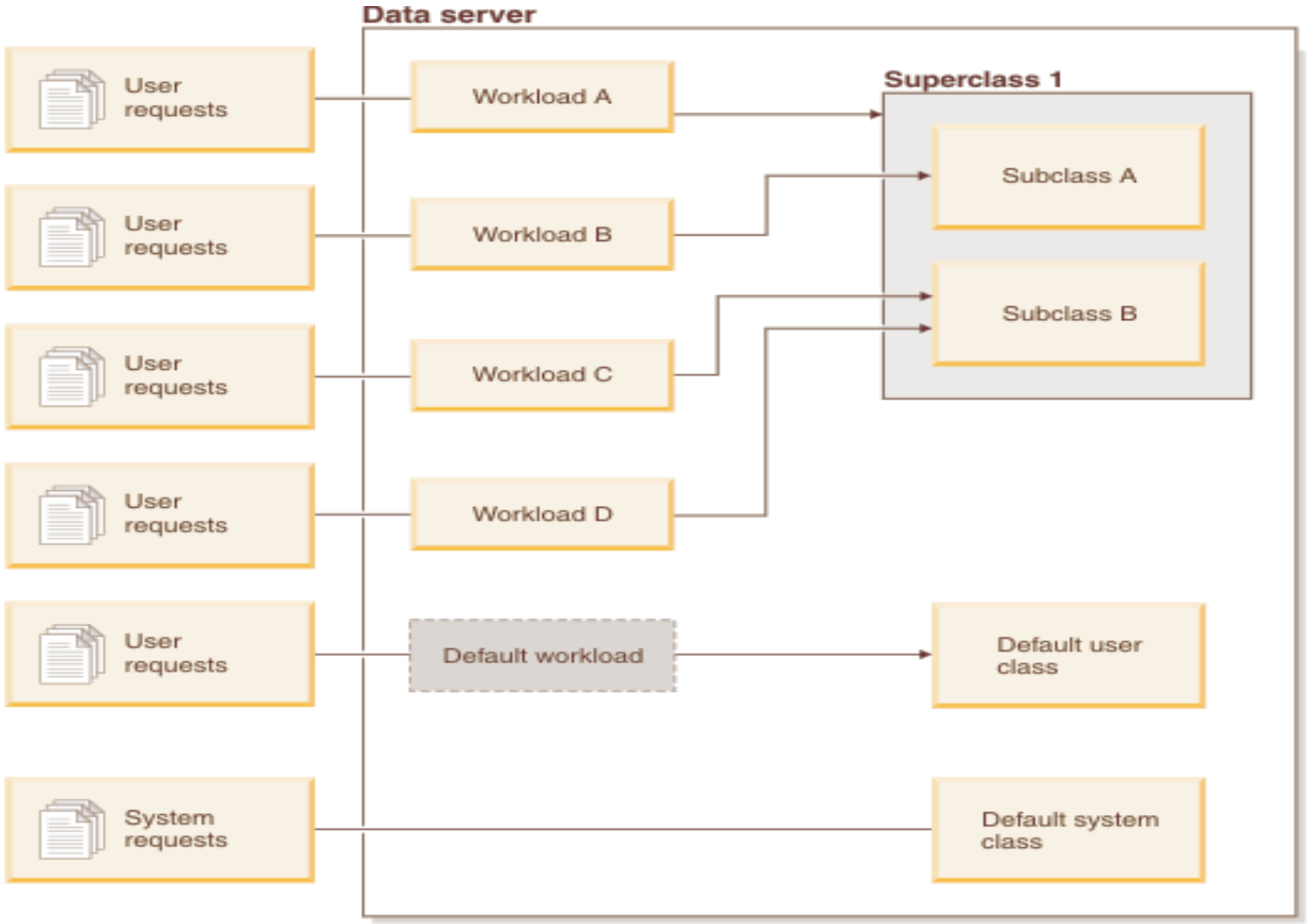
WLM Features for Stage 2: Management.

- **New database objects are provided to implement management:**
  - SERVICE CLASS

WLM Features – Service Class

- **Definition of the 'execution environment' for work on the data server**
  - Resource assignment
  - Monitoring
  - Workload control
- **Resources explicitly controlled include:**
  - Agent priority (CPU)
  - Prefetch priority (I/O)
  - External WLM tag: AIX Workload Manager.
- **2 tiers of service classes are allowed:**
  - Superclass
  - Subclass
- **Three default service class (ponerlas explícitamente)**
  - Sysdefaultuserclass
  - Sysdefaultsystemclass
  - Sysdefaultmaintenanceclass

WLM Features – Workload Mapping



WLM Features – Service Class. Examples.

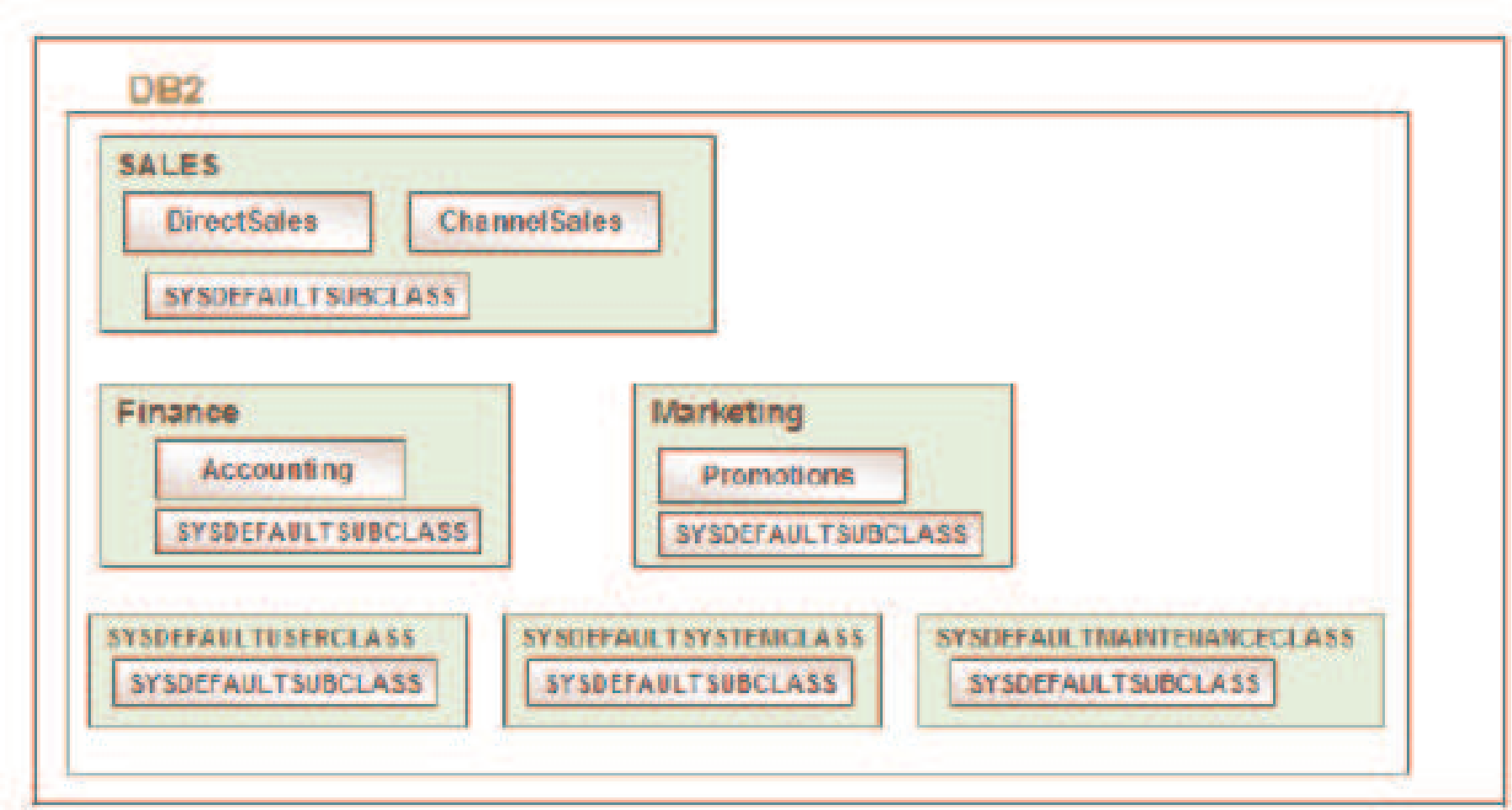
- **Create service superclasses**

```
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;
```

WLM Features – Service Class



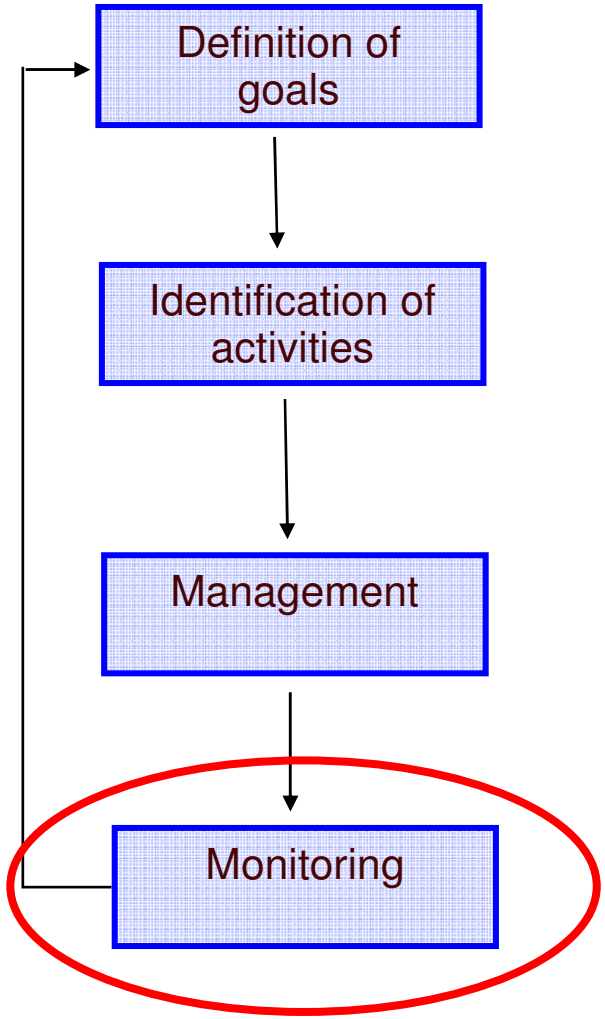
WLM Feature - Threshold

- **Sometimes work arrives that exceeds normal expectations**
  - Individual activities consume excessive resources
  - Arrival rate of work is too high
- **DB2 Thresholds can be defined to recognize these exceptions:**
  - Elapsed time
  - Idle time
  - Estimated cost
  - Rows returned
  - Temporary space used
  - Concurrent workload occurrences
  - Total database connections
  - Total service class connections
  - Concurrent database activities
  - Concurrent workload activities

WLM Feature – Threshold (cont.)

- **If a threshold is violated, the following actions can be taken:**
  - Collect data
    - Useful for future analysis on the activity causing the violation
  - Stop execution
    - Do not proceed with execution
  - Continue execution
    - Perhaps logging the violation is sufficient
  - Queuing of activities will result if concurrent service class connections or concurrent database activity thresholds are exceeded.

Stages of Workload Management . WLM FEATURES



WLM Feature - Monitoring

- **Monitoring is critical to determine is business objectives are being met and problem determination**

- **Two types of monitoring available:**
  - Real-time monitoring
    - Light-weight access to in memory information on workload objects
  - Historical monitoring
    - Event monitor

WLM Feature – Real-time monitoring

- **Many of the workload information and statistics are stored in memory and accessible via lightweight table functions**

WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES
- Returns a list of workload occurrences in the specified service class

WLM_GET_SERVICE_CLASS_AGENTS
- Returns a list of agents that currently exist in the specified service class

WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES
- Returns a list of all activities for the given application on the specified partition which have not yet completed

WLM_GET_ACTIVITY_DETAILS
- Returns detailed information about a specific activity

WLM_GET_SERVICE_SUPERCLASS_STATS
- Returns basic statistics of one or more service super classes

WLM_GET_SERVICE_SUBCLASS_STATS
- Returns basic statistics of one or more service subclasses

WLM_GET_WORKLOAD_STATS
- Returns basic statistics of one or more workloads

WLM Feature – Historical monitoring

- **Three new event monitor:**

ACTIVITY MONITOR

- This monitor captures information about individual activities in a service class, workload, or work class or activities that violated a threshold.

THRESHOLD VIOLATIONS EVENT MONITOR

- This monitor captures information when a threshold is exceeded. It indicates what threshold was exceeded, the activity that was the source of the exception, and what action was taken when it occurred (i.e. was the activity stopped or allowed to continue).

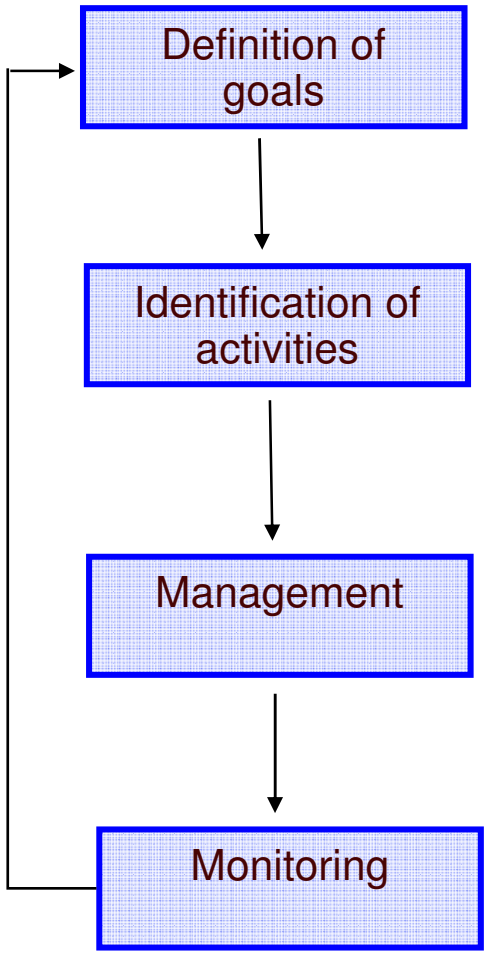
STATISTICS EVENT MONITOR

- This monitor serves as a low-overhead alternative to capturing detailed activity information by collecting aggregate data (for example, the number of activities completed and average execution time).

Agenda

- Workload Manager. Reasons and Objectives.
- Workload Manager. Stages and Features.
- **Best Practices and Examples.**
- Summary

WLM. Best Practices. 1.- Monitor to identify



- Obtain work characteristics of your environment before customize WLM.

- Characteristics of interest:
  - User Names
  - Applications Names
  - Concurrency levels
  - Resource Consumption

WLM. Best Practices. 1.- Monitor to identify. Examples

- *USE STATISTICAL DATA*

CREATE EVENT MONITOR DB2STATISTICS FOR STATISTICS WRITE TO TABLE;

SET EVENT MONITOR DB2STATISTICS STATE 1;

ALTER SERVICE CLASS SYSDEFAULTSUBCLASS UNDER SYSDEFAULTUSERCLASS COLLECT AGGREGATE ACTIVITY DATA EXTENDED;

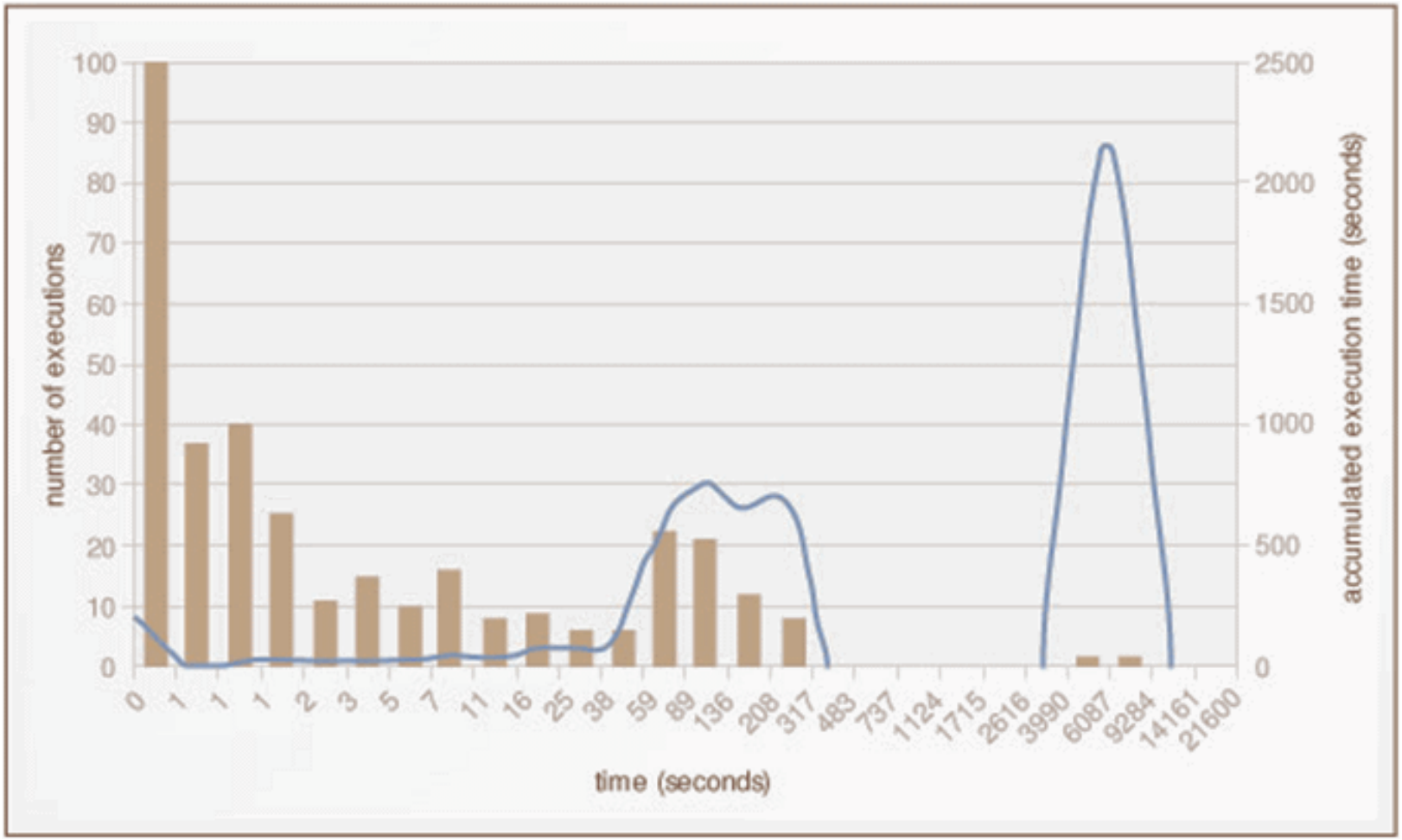
Run a typical workload to collect some statistics.

CALL WLM_COLLECT_STATS;

SELECT TOP/1000 AS TIME_seconds, SUM(NUMBER_IN_BIN) AS #EXECUTIONS FROM HISTOGRAMBIN_DB2STATISTICS WHERE HISTOGRAM_TYPE = 'CoordActExecTime' GROUP BY TOP/1000 order by TOP/1000 ;

TIME_SECONDS	#EXECUTIONS
0	385
1	37
1	40
1	25
2	11
3	15
5	10
7	16
11	8
16	9
25	6
38	6
59	22
89	21
136	12
208	8
317	0
483	0
737	0
1124	0
1715	0
2616	0
3990	1
6086	1
9283	0
14160	0
21600	0

WLM. Best Practices. 1.- Monitor to identify.



WLM. Best Practices. 1.- Monitor to identify

2. USE ADMIN VIEWS.

```
SELECT A.DBPARTITIONNUM, COUNT(*) #CONNECTION,
     INTEGER(SUM(AGENT_USR_CPU_TIME_S+AGENT_SYS_CPU_TIME_S))
     CPU_SECOND,SUBSTR(APPL_NAME,1,20) APPLNAME,
SUBSTR(PRIMARY_AUTH_ID,1,16) SYSTEM_USER
FROM SYSIBMADM.SNAPAPPL_INFO A, SYSIBMADM.SNAPAPPL B WHERE APPL_NAME
     NOT IN ('db2stmm','db2wlmd','db2taskd','db2evmg_DB2DETAILDEA')
AND A.AGENT_ID=B.AGENT_ID
AND A.DBPARTITIONNUM=B.DBPARTITIONNUM
GROUP BY APPL_NAME,PRIMARY_AUTH_ID, A.DBPARTITIONNUM
ORDER BY A.DBPARTITIONNUM,APPL_NAME;
```

```
DBPARTITIONNUM #CONNECTION CPU_SECOND  APPLNAME            SYSTEM_USER
-------------- ----------- ----------- ------------------- -------------
             0         504         312 db2batch            DB2INST3
             0           1           0 db2bp               DB2INST3
             1         496        3868 db2batch            DB2INST3
             1           1           0 db2bp               DB2INST3
     ...
            16         497        3681 db2batch            DB2INST3
            16           1           0 db2bp               DB2INST3
```

WLM. Best Practices. 1.- Monitor to identify

3. OTHER OPTIONS TO MONITOR
   - Performance Expert
   - Activity Event Monitor.

WLM. Best Practices. 2.- Determine the connection atributes

- Determine the connection attributes required to define workload objects.

```
SELECT COUNT(*) COUNT, SUBSTR(APPLICATION_NAME, 1, 10) APPLNAME,
SUBSTR(SYSTEM_AUTH_ID,1,10) SYSTEM_USER , SUBSTR(SESSION_AUTH_ID,1,10)
SESSION_ID , SUBSTR(CLIENT_USER,1,10) CLIENT_USER,
SUBSTR(CLIENT_WRKSTNNAME,1,21) CLIENT_WRKSTNNAME ,
SUBSTR(CLIENT_ACCTNG,1,10) CLIENT_ACCTNG, SUBSTR(CLIENT_APPLNAME,1,10)
CLIENT_APPLNAME FROM
TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('', '', -2)) A GROUP BY
APPLICATION_NAME, SYSTEM_AUTH_ID, SESSION_AUTH_ID , CLIENT_WRKSTNNAME,
CLIENT_ACCTNG, CLIENT_USER, CLIENT_APPLNAME;
```

```
COUNT  APPLNAME    SYSTEM_USER      SESSION_ID CLIENT_USER CLIENT_WRKSTNNAME    CLIENT_ACCTNG CLIENT_APPLNAME
------ --------- ---------------    ---------- ----------- -------------------- ------------- --------------
   501  db2batch     DB2INST3         DB2INST3         -           -                    -             -
     1 db2bp        DB2INST3         DB2INST3         nela        appl#1.torolab.ibm.c 123-456       boss
```

WLM. Best Practices. 3. Classify

Classify the applications and users into different group according to their type and business priority.

EXAMPLE: Data Warehouse environment.

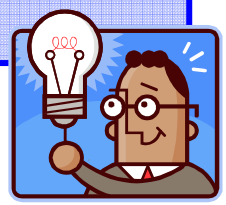
- Daily reporting queries
- Ad hoc or complex queries
- ETL Jobs for real time warehouses

WLM. Best Practices. 4. Create WLM objects. Service class

- **Create a service class with three subclasses to determine resource allocation**.

CREATE SERVICE CLASS TEST_CLASS;
CREATE SERVICE CLASS HIGH_PRIO UNDER TEST_CLASS;
CREATE SERVICE CLASS MEDIUM_PRIO UNDER TEST_CLASS;
CREATE SERVICE CLASS LOW_PRIO UNDER TEST_CLASS;

Start with one service superclass
Create subclasses for low, medium and high cost work



WLM. Best Practices. 4. Create WLM objects. Service Class. Priority settings.

- **Assign the priority in the service subclasses.**

ALTER SERVICE CLASS HIGH_PRIO UNDER TEST_CLASS
AGENT PRIORITY -20 PREFETCH PRIORITY HIGH;

ALTER SERVICE CLASS MEDIUM_PRIO UNDER TEST_CLASS
AGENT PRIORITY DEFAULT PREFETCH PRIORITY MEDIUM;

ALTER SERVICE CLASS LOW_PRIO UNDER TEST_CLASS
AGENT PRIORITY 20 PREFETCH PRIORITY LOW;

- **Ensure that system work take precedence over user work.**

ALTER SERVICE CLASS SYSDEFAULTSYSTEMCLASS
AGENT PRIORITY -20 PREFETCH PRIORITY HIGH;

WLM. Best Practices. 4. Create WLM objects. WORKLOAD

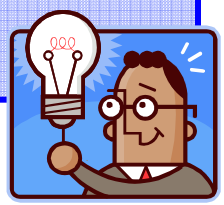
- **Create three workloads that indentify the more and the less important applications or users, and assign the workloads to the service subclasses.**

CREATE WORKLOAD HIGH_PRIO APPLNAME ('db2bp') SYSTEM USER ('DB2INST3')
SERVICE CLASS HIGH_PRIO;

CREATE WORKLOAD MEDIUM_PRIO APPLNAME ('dss.exe')
SERVICE CLASS MEDIUM_PRIO;

CREATE WORKLOAD LOW_PRIO APPLNAME ('db2batch')
SERVICE CLASS LOW_PRIO;

Start by mapping applications to business priorities



WLM. Best Practices. 5. Create WLM objects. work class /work action

- **Distinguish between types of work using work actions.**
- **Assign the workload to a service superclass.**

CREATE WORKLOAD ALL_PRIO APPLNAME ('db2bp') *SYSTEM USER ('DB2INST3') SERVICE CLASS TEST;*

- **Create a work class set to define work type criteria**

CREATE WORK CLASS SET control_cost
(WORK CLASS long WORK TYPE READ FOR TIMERONCOST FROM 2000001 To UNBOUNDED,
WORK CLASS medium WORK TYPE READ FOR TIMERONCOST FROM 20001 TO 2000000,
WORK CLASS short WORK TYPE READ FOR TIMERONCOST FROM 0 TO 20000);

- **Create the work actions to map the work to the HIGH, MEDIUM and LOW priority subclasses.**

CREATE WORK ACTION SET query_cost FOR SERVICE CLASS POWER USING WORK CLASS SET control_cost
(WORK ACTION MAP_LONG ON WORK CLASS long MAP ACTIVITY TO LOW_PRIO,
WORK ACTION MAP_MEDIUM ON WORK CLASS medium MAP ACTIVITY TO MEDIUM_PRIO
WORK ACTION MAP_SHORT ON WORK CLASS short MAP ACTIVITY TO HIGH_PRIO);

WLM. Best Practices. 6. Thresholds. PROTECT YOUR SYSTEM FROM BEING OVERLOADED

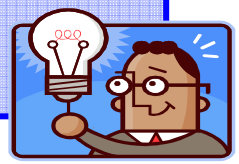
- **Concurrency control**

- **Example: Permit a maximum of 20 concurrent complex queries.**

```
CREATE THRESHOLD QUEUE_LOW_PRIO
FOR SERVICE CLASS LOW_PRIO UNDER CLASS_TEST
 ACTIVITIES ENFORCEMENT DATABASE
WHEN CONCURRENTDBCOORDACTIVITIES > 20 CONTINUE;
```

WLM. Best Practices. 6. Thresholds. PREVENT MONSTER QUERIES

Queries with a very high cost can be controlled either predictively before query evaluation starts, or reactively in response to how the query is behaving during execution.



- **Predictive threshold.**

```
CREATE THRESHOLD HIGH_COSTS FOR SERVICE CLASS LOW_PRIO
UNDER CLASS_TEST
ACTIVITIES ENFORCEMENT DATABASE
WHEN ESTIMATEDSQLCOST > 50000000
COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
STOP EXECUTION;
```

- **Reactive threshold.**

```
CREATE THRESHOLD TOO_LONG FOR SERVICE CLASS CLASS_TEST
ACTIVITIES ENFORCEMENT DATABASE
WHEN ACTIVITYTOTALTIME > 60 MINUTES
COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
STOP EXECUTION;
```

Reactive thresholds:

- SQLRowsReturned
- SQLTempSpace
- ConnectionIdleTime

WLM. Best Practices. 6. Thresholds.
LIMIT THE NUMBER OF CONCURRENT LOAD OPERATIONS

- **Control load operations by putting the load work type into a separate work class.**

CREATE WORK CLASS SET LOAD_TYPE
(WORK CLASS LOAD_WC WORK TYPE LOAD);

- **To limit the number of concurrent load operations to one at the database level, create a work action:**

CREATE WORK ACTION SET CONTROL_LOAD FOR DATABASE
USING WORK CLASS SET LOAD_TYPE
(WORK ACTION LIMIT_LOAD ON WORK CLASS LOAD_WC
WHEN CONCURRENTDBCOORDACTIVITIES > 1
CONTINUE);

WLM. Best Practices. 6. Thresholds.
LIMIT THE NUMBER OF CONCURRENT LOAD OPERATIONS (cont)

- **Review the state of the load operations using the monitor table WLM_GET_SERVICE_CLASS_WORKLOAD_OCURRENCES.**

```
SELECT SUBSTR(APPLICATION_NAME,1,10) AS APPL_NAME,
SUBSTR(CHAR(APPLICATION_HANDLE),1,10) AGENTID,
SUBSTR(WORKLOAD_NAME,1,22) AS WORKLOAD_NAME, SUBSTR(CLIENT_APPLNAME,1,25)
AS CLIENT_APPLNAME, WORKLOAD_OCCURRENCE_STATE AS WL_STATE
FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('','',-2))
ORDER BY WORKLOAD_OCCURRENCE_STATE DESC;
```

```
APPL_NAME  AGENTID    WORKLOAD_NAME          CLIENT_APPLNAME            WL_STATE
---------- ---------- ---------------------- -------------------------- ----------
db2bp           65638 SYSDEFAULTUSERWORKLOAD CLP load_from_flat3.sql    UOWWAIT
db2bp           65604 HIGH_PRIO              -                          UOWEXEC
db2bp           65637 SYSDEFAULTUSERWORKLOAD CLP load_from_flat2.sql    QUEUED
db2bp           65639 SYSDEFAULTUSERWORKLOAD CLP load_from_flat1.sql    QUEUED
db2bp           65640 SYSDEFAULTUSERWORKLOAD CLP load_from_flat4.sql    QUEUED
db2bp           65660 HIGH_PRIO              CLP load_from_flat.sql     QUEUED
```

Agenda

- Workload Manager. Reasons and Objectives.
- Workload Manager. Stages and Features.
- Best Practices and Examples.
- **Summary**

Summary of Key WLM Features

- **DB2 Service Class**
  - Serves as the primary point of resource control for executing work
  - Acts as point of integration with AIX WLM for work being done within database
- **DB2 Workload**
  - Serves as the primary point of control for submitters of work
  - Acts as primary router of work to a specific DB2 Service Class
- **DB2 Threshold**
  - Provides limits to control behaviours of database activities based on predictive and reactive elements
  - Provides limits to control rate of concurrency for database activities
- **DB2 Work Action Set**
  - Provides ability to discriminate between different types of database activities for service subclass mapping or for DB2 Threshold assignment
- **DB2 WLM Monitor and Control capabilities**
  - New table functions, event monitors, and stored procedures to provide monitoring and control mechanisms for DB2 WLM

Summary (1 of 3)

- **Start with solid DB design, well tuned queries**
- **Evaluate your business commitments and priorities to set goals before you start**
- **Use tools in Data Studio and Performance Expert**
- **Divide and conquer work based on priority**
  - Create a workload definition for each source of work
  - Create a service subclass for each category of work
- **Think globally, act locally**
  - Consider your system holistically
  - Apply thresholds at the subclass or workload level

Summary (2 of 3)

- **Assign activities to subclasses**
  - Via workload definitions when appropriate.
  - Via work actions and estimated costs to prioritize by cost.

- **Concurrency**
  - Limit concurrency in low and medium priority subclasses.
  - Use an ActivityTotalTime threshold in conjunction with concurrency thresholds as a failsafe against deadlock.

Summary (3 of 3)

- **Work iteratively**
  - Monitor first, to collect a baseline
  - Apply controls individually
  - Evaluate the effect of each change before moving on
  - Initially create thresholds as monitoring only
    - Collect activity data for threshold violations
    - Validate thresholds by examining the violations
    - Convert from CONTINUE -> STOP only after validated
- **Monitoring**
  - Leave low overhead monitoring turned on permanently
  - Performance Expert manages WLM statistics event monitors for you
  - If you need to manage your own event monitors
    - Keep monitoring data for later comparison
    - Prune event monitors

Further Reading

- **Best practices :**
- **http://www.ibm.com/developerworks/wikis/display/data/Best+Practice+-+DB2+Workload+Management**
- **Tutorial for workload management:**
- **http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.wlm.doc/doc/c0053139.html**
- **IBM DB2 Database for Linux, UNIX, and Windows Information Center: http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp**
- ***WLM FAQ in DB2 9.5 documentation: http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.wlm.doc/doc/c0052604.html***
- ***WLM Hands On Tutorial in DB2 9.5 documentation and available for download on IBM Developer Works:***
- ***http://www-28.ibm.com/developerworks/forums/servlet/JiveServlet/download/1116-179878-14005115-301960/wlmiodlab.zip***