Using Database Resource Manager

ORACLE

Objectives

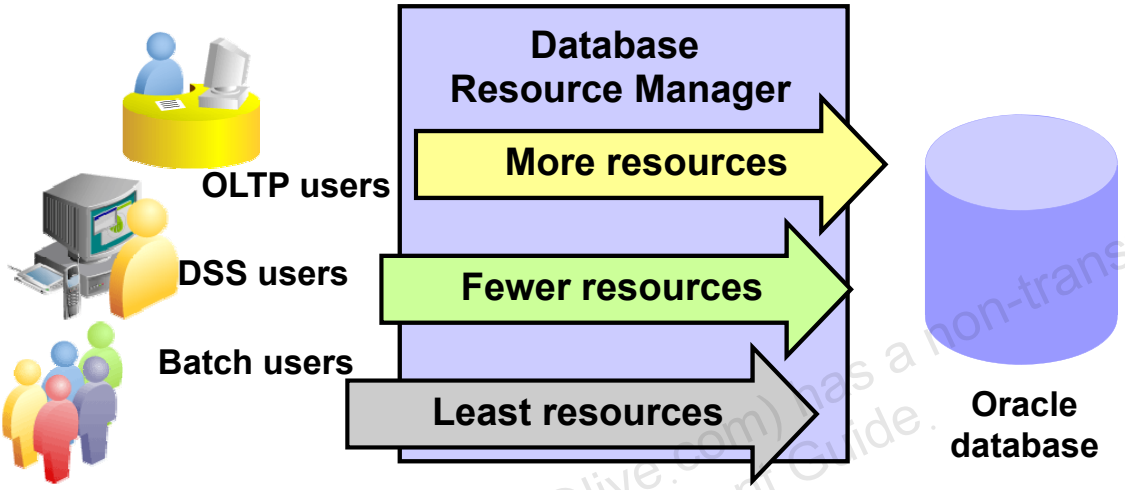
After completing this lesson, you should be able to do the following:

- Configure the Database Resource Manager
- Access and create resource plans
- Create consumer groups
- Specify directives for allocating resources to consumer groups
- Map consumer groups to plans
- Activate a resource plan
- Monitor the Resource Manager

Database Resource Manager: Overview

Use the Resource Manager to:

- Manage mixed workload
- Control system performance



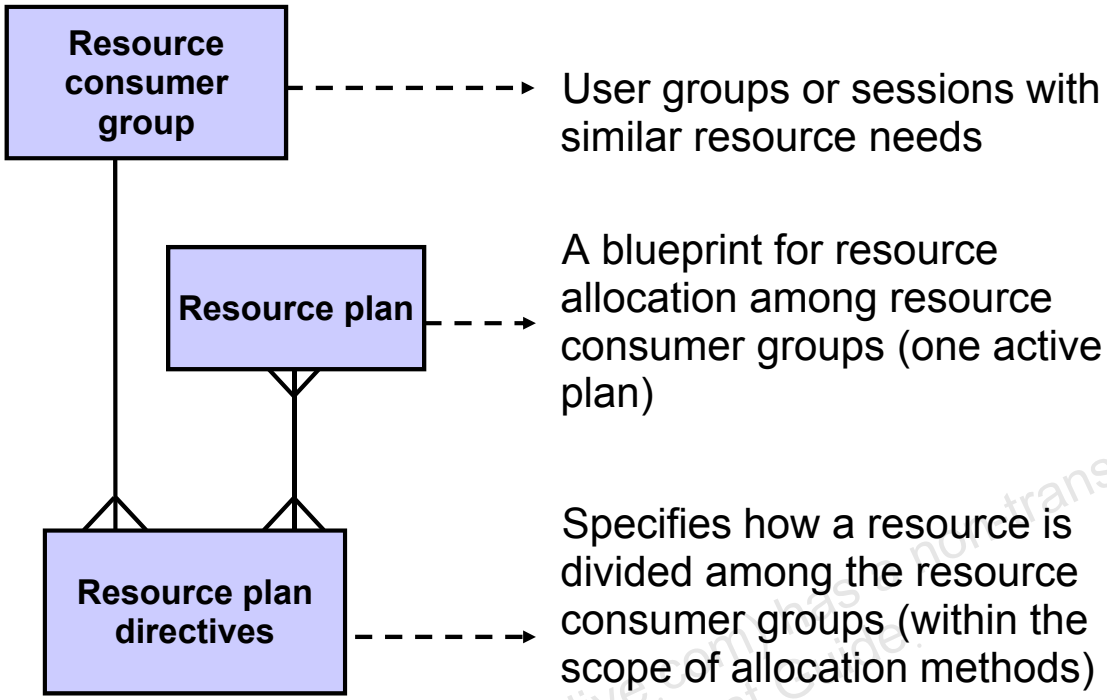
By using the Database Resource Manager (also called the Resource Manager), you have more control over the allocation of machine resources than is normally possible through operating system resource management alone. If resource management decisions are made by the operating system, it can lead to problems such as:

- Excessive overhead resulting from operating system context switching of Oracle Database server processes when the number of server processes is high
- Suspension of a database server process that is holding a latch
- Unequal distribution of resources among all Oracle Database processes, and an inability to prioritize one task over another
- Inability to manage database-specific resources, such as parallel execution servers and active sessions

The Database Resource Manager controls the distribution of resources among various sessions by controlling the execution schedule inside the database. By controlling which sessions run and for how long, the Database Resource Manager can ensure that resource distribution matches the plan directive and, therefore, the business objectives. With the Database Resource Manager, you can guarantee groups of users a minimum amount of processing resources regardless of the load on the system and the number of users.

The `DBMS_RESOURCE_MANAGER_PRIVS` package contains the procedures to grant and revoke the `ADMINISTER_RESOURCE_MANAGER` system privilege, which is a prerequisite for invoking the Resource Manager.

Database Resource Manager: Concepts



Administering systems by using the Database Resource Manager involves the use of resource plans, resource consumer groups, and resource plan directives.

A *resource consumer group* defines a set of users or sessions that have similar requirements for using system and database resources.

A *resource plan* specifies how the resources are distributed among various resource consumer groups. The Database Resource Manager also allows for creation of plans within plans, called *subplans*.

Resource plan directives specify how a particular resource is shared among consumer groups or subplans. You associate resource consumer groups and subplans with a particular resource plan through plan directives.

Resource allocation methods determine what policy to use when allocating for any particular resource. Resource allocation methods are used by resource plans and resource consumer groups.

Using the Resource Manager

- You can manage database and operating system resources, such as:
  - CPU usage
  - Degree of parallelism
  - Number of active sessions
  - Undo generation
  - Operation execution time
  - Idle time
  - Database consolidation
  - Server consolidation
- You can also specify criteria that, if met, cause the automatic switching of sessions to another consumer group.

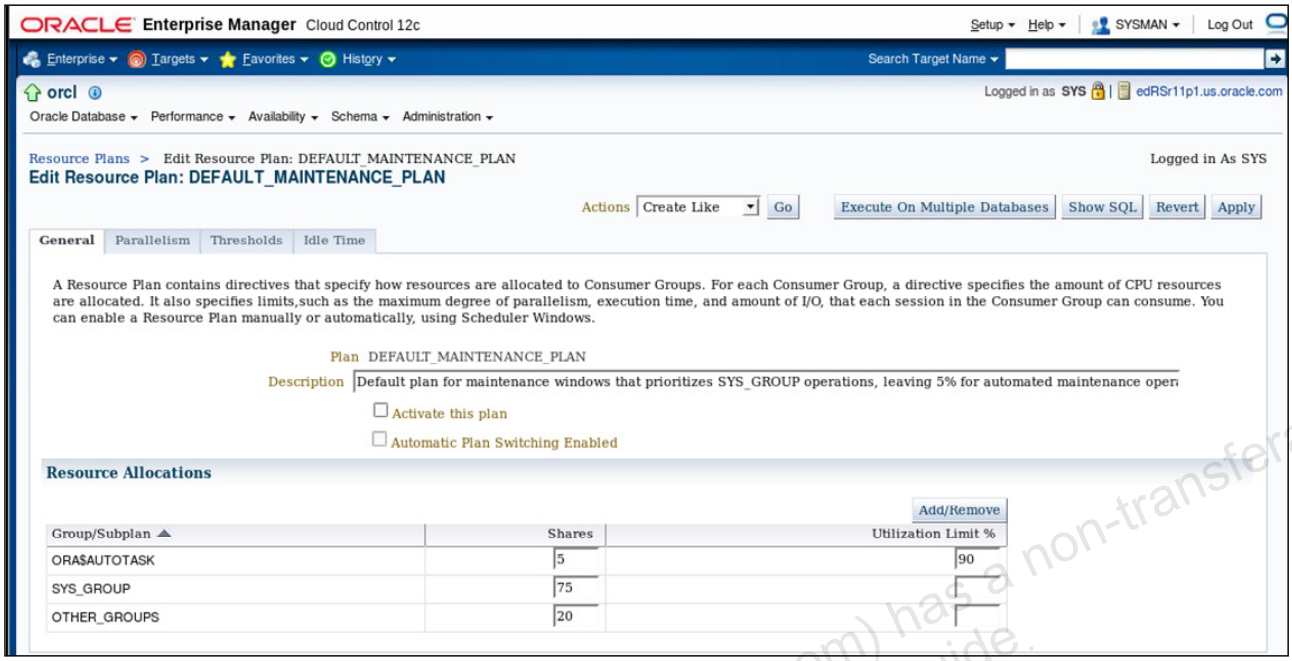
The Database Resource Manager provides several means of allocating resources:

- **CPU Method:** Enables you to specify how CPU resources are allocated among consumer groups and subplans
- **Degree of Parallelism Limit:** Enables you to control the maximum degree of parallelism for any operation within a consumer group
- **Active Session Pool with Queuing:** Allows you to limit the number of concurrent active sessions for a consumer group or subplan. If a group exceeds the maximum allowed number of sessions, new sessions are placed in a queue where they wait for an active session to complete. You can also specify a time limit on how long a session will wait before exiting with an error.
- **Undo Pool:** Enables you to control the total amount of undo that can be generated by a consumer group or subplan. Whenever the total undo space exceeds the amount specified by `UNDO_POOL`, no further `INSERT`, `UPDATE`, or `DELETE` commands are allowed until undo space is freed by another session in the same group or the undo pool is increased for the consumer group. If the consumer group's quota is exceeded during the execution of a DML statement, the operation aborts and returns an error. Queries are still allowed, even if a consumer group has exceeded its undo threshold.

- **Execution Time Limit:** Allows you to specify a maximum execution time allowed for an operation. The Oracle Database server uses cost-based optimizer statistics to estimate how long an operation will take. If it is longer than the maximum time allowed (`MAX_EST_EXEC_TIME`), the operation returns an error and is not started. If a resource consumer group has more than one plan directive with `MAX_EST_EXEC_TIME` specified, the Resource Manager chooses the most restrictive of all incoming values.
- **Idle Time Limit:** Enables you to specify an amount of time for which a session can be idle, after which it will be terminated (`MAX_IDLE_TIME`). You can further restrict the Resource Manager to terminate only those sessions that are blocking other sessions (`MAX_IDLE_TIME_BLOCKER`).
- **Consumer Group Switching:** Specifies conditions that will cause a session to switch consumer groups. Typically, overuse of a resource is specified and a session is switched to a more restrictive consumer group. The session remains in the switched consumer group until it becomes idle, or if directed after the top-level call is completed. Then it will return to the initial consumer group. The initial consumer group is the group that a session is assigned to at login. The top is the current PL/SQL block or each SQL statement that is issued outside a PL/SQL block by the client. You can create a plan directive, so that the Resource Manager automatically switches the user back to the initial consumer group at the end of the top call.
- **Database Consolidation:** The Resource Manager enables you to optimize resource allocation among concurrent database sessions. Database consolidation requires that applications are isolated from each other. If one application experiences an increase in workload, that increase should not affect other applications. In addition, the performance of each application should be consistent. Good candidate applications for database consolidation are automated maintenance tasks because currently, these applications can take up to 100% of the server CPU resources.
- **Server Consolidation:** Because many test, development and small production databases are unable to fully utilize the servers that they are on, *server consolidation* provides a possible alternative. With server consolidation, resources are more fully utilized by running multiple database instances on the server. The method for managing CPU allocations on a multi-CPU server with multiple database instances is called Instance Caging. Because Instance Caging is simple to configure and does not require any new software to be licensed or installed, it is an excellent alternative to other server consolidation tools, such as virtualization and O/S workload managers.

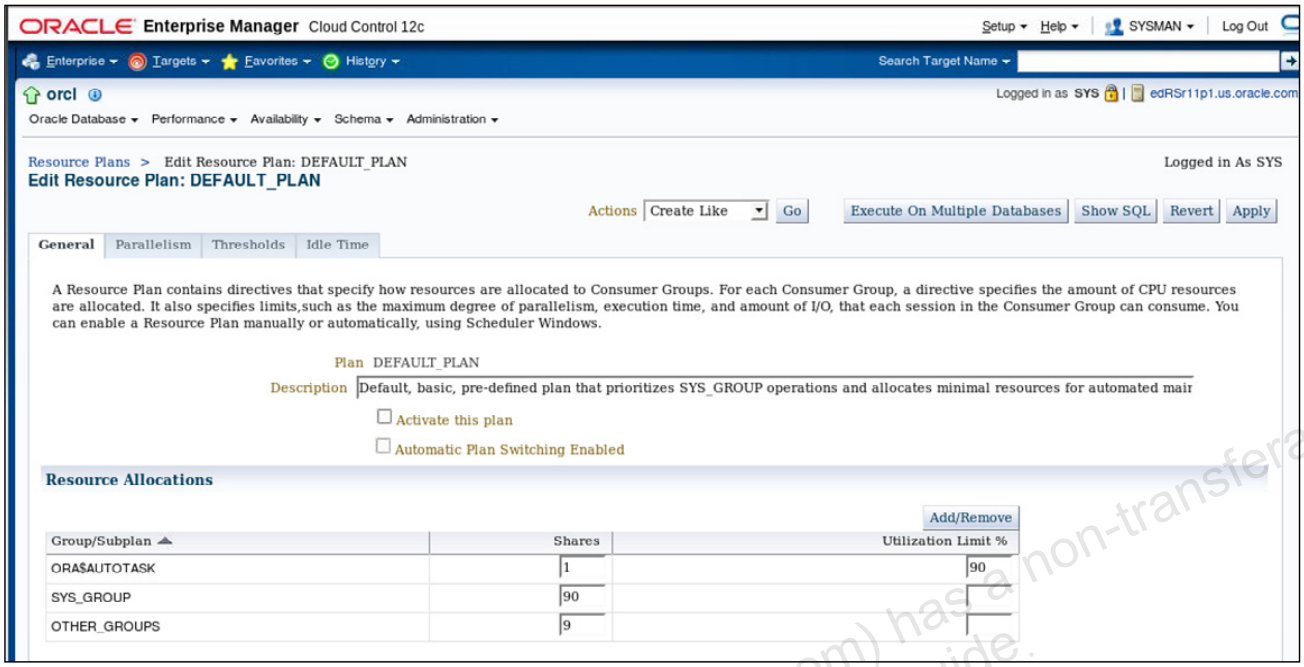
You can access resource plans with the graphical interface of Enterprise Manager Cloud Control or the command line of the `DBMS_RESOURCE_MANAGER` package.

Default Plan for Maintenance Windows



The automated maintenance tasks rely on the Resource Manager being enabled during the maintenance windows. When a maintenance window opens, the `DEFAULT_MAINTENANCE_PLAN` resource manager plan is automatically set to control the amount of CPU used by automated maintenance tasks. To be able to give different priorities to each possible task during a maintenance window, various consumer groups are assigned to `DEFAULT_MAINTENANCE_PLAN`.

Default Plan



The DEFAULT_PLAN resource plan is one of the default plans provided for you. It contains directives for the following provided consumer groups:

- **SYS_GROUP:** The initial consumer group for the SYS and SYSTEM users
- **OTHER_GROUPS:** Used for all sessions that belong to consumer groups that are not part of the active resource plan. There must be a plan directive for OTHER_GROUPS in any active plan.
- **ORA$AUTOTASK:** A group with lower priority than SYS_GROUP and OTHER_GROUPS in this plan

The initial consumer group of a user is the consumer group to which any session created by that user initially belongs. If you have not set the initial consumer group for a user, the user's initial consumer group will automatically be DEFAULT_CONSUMER_GROUP.

The DEFAULT_PLAN and associated resource consumer groups can be used or not used. It can be a template for new resource plans; it can be modified or deleted. Use it as appropriate for your environment.

Creating a Simple Resource Plan

Create consumer groups and allocate resources to them by executing a single procedure call:

```
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_SIMPLE_PLAN(SIMPLE_PLAN =>
    'SIMPLE_RESPLAN1',
CONSUMER_GROUP1 => 'CONSGROUP1', GROUP1_PERCENT => 80,
CONSUMER_GROUP2 => 'CONSGROUP2', GROUP2_PERCENT => 20);
END;
```

Consumer Group	Level 1	Level 2	Level 3
SYSGROUP	100%		
CONSGROUP1		80%	
CONSGROUP2		20%	
OTHER_GROUPS			100%

ORACLE

You can create a simple resource plan by using the `DBMS_RESOURCE_MANAGER.CREATE_SIMPLE_PLAN` procedure. You can create consumer groups and allocate resources to them by using this single procedure.

The `CREATE_SIMPLE_PLAN` procedure accepts the following arguments:

- `SIMPLE_PLAN`: Name of the plan
- `CONSUMER_GROUP1` (through `CONSUMER_GROUP8`): Consumer group name(s)
- `GROUP1_PERCENT` (through `GROUP8_PERCENT`): CPU resource allocated to the group(s)

The plan uses the default CPU allocation policy (`EMPHASIS`). Each consumer group uses the default scheduling policy (`ROUND_ROBIN`).

The consumer groups specified in the plan are allocated CPU percentage at level 2 as shown in the table in the slide. The plan also includes the `SYS_GROUP` consumer group and the `OTHER_GROUPS` consumer group. As shown in the table in the slide, `SYSGROUP` is allocated 100% of the CPU at level 1 and `OTHER_GROUPS` is allocated 100% of the CPU at level 3.

Creating a Complex Resource Plan

1. Create a pending area.
2. Create, modify, or delete consumer groups.
3. Map sessions to consumer groups.
4. Create the resource plan.
5. Create resource plan directives.
6. Validate the pending area.
7. Submit the pending area.

You can perform the following steps by using the named procedures in the `DBMS_RESOURCE_MANAGER` package or by using Enterprise Manager Cloud Control.

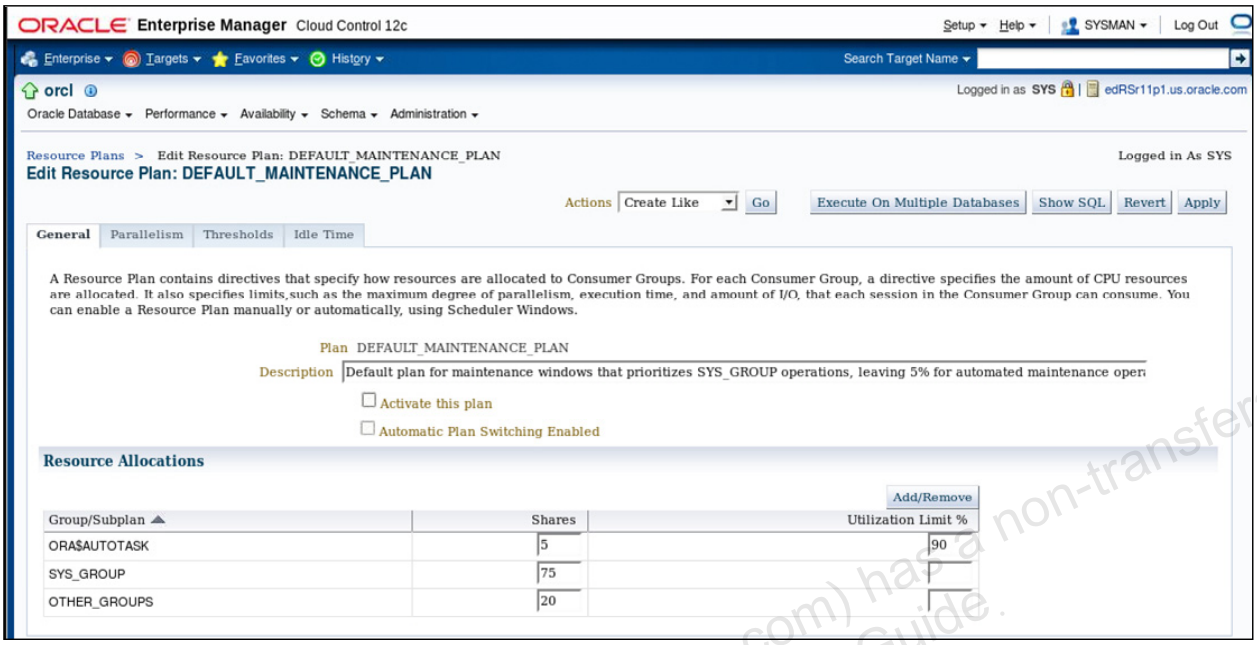
1. Create a pending area: To create a new resource plan, or update or delete an existing resource plan, you must create a pending area. The pending area is a staging area where you can create and modify plans without affecting executing applications. After you create the pending area, the Oracle Database server copies existing plans into it so they can be updated if required. Create a pending area by using the `CREATE_PENDING_AREA` procedure.
2. Create, modify, or delete consumer groups: Create a resource consumer group and specify a resource allocation method (`ROUND-ROBIN` or `RUN-TO-COMPLETION`) for distributing CPU among the sessions in the consumer group. Use the `CREATE_CONSUMER_GROUP` procedure.
3. Map sessions to consumer groups: Use the `SET_CONSUMER_GROUP_MAPPING` procedure to map a session attribute type and attribute value to a consumer group. The session attribute types include Oracle Database username, database service name, operating system username, client program name, and module name. Refer to the *Oracle Database Administrator's Guide* for a complete list of accepted session attributes.

Note: Unassigned sessions are part of the `OTHER_GROUPS` consumer group.

4. Create the resource plan: When you create the resource plan, you provide a name and optionally specify the resource allocation method for CPU (`EMPHASIS` or `RATIO`), active session pool resource allocation method (`ACTIVE_SESS_POOL_ABSOLUTE`), resource allocation method for degree of parallelism (`PARALLEL_DEGREE_LIMIT_ABSOLUTE`), and queuing resource allocation method (`FIFO_TIMEOUT`). Use `CREATE_PLAN` to create a resource plan. Additional information follows in this lesson.
5. Create resource plan directives: Resources are allocated to consumer groups based on the resource plan directives. Through resource plan directives, you can specify:
   - The maximum number of concurrently active sessions for a consumer group
   - A limit on the degree of parallelism for any operation
   - The time (in CPU seconds) that a call can execute before an action is taken
   - A maximum in kilobytes (K) on the total amount of undo for uncommitted transactions that can be generated by a consumer group

   Use `CREATE_PLAN_DIRECTIVE` to specify resource plan directives.
6. Validate the pending area: Use the `VALIDATE_PENDING_AREA` procedure to validate the pending area at any time. The validate procedure checks for the following:
   - Plans do not contain loops
   - All plans and resource consumer groups referred to by plan directives exist
   - All plans have plan directives that point to either plans or resource consumer groups
   - All percentages in any given level do not add up to greater than 100
   - A plan that is currently being used as a top plan by an active instance is not being deleted
   - That certain parameters appear only in plan directives that refer to resource consumer groups
   - No more than 28 resource consumer groups appear in any active plan
   - Plans and resource consumer groups do not have the same name
   - A plan directive for `OTHER_GROUPS` appears somewhere in any active plan
7. Submit the pending area: After validating the pending area, submit it by using `SUBMIT_PENDING_AREA`. When you submit the pending area, new and updated plan information is stored in the data dictionary. New plans are not activated when the pending area is submitted. Modified plans are reactivated with their new plan definition.

Specifying Resource Plan Directives



In Enterprise Manager Cloud Control, there are several property pages that you can use to specify plan directives:

1. On the General page, associate consumer groups with plans and specify how much CPU each consumer group or subplan is allocated.
2. On the Parallelism page, specify a limit on the degree of parallelism for any operation issued by this consumer group, a limit on the total number of parallel server processes that can be used by all sessions in this consumer group, and the maximum time a parallel statement can be queued.
3. On the Thresholds page, specify the time duration or the resource limits under which a session can execute in a consumer group.
4. On the Idle Time page, specify an amount of time that a session can be idle, after which it will be terminated. You can further restrict such termination to only those sessions that are blocking other sessions.

Resource Allocation Methods for Resource Plans

Parameter	Possible Values
MGMT_MTH: Allocating CPU usage	EMPHASIS, RATIO
PARALLEL_DEGREE_LIMIT_MTH: Limiting degree of parallelism of any operation	PARALLEL_DEGREE_LIMIT_ABSOLUTE
ACTIVE_SESS_POOL_MTH: Limiting number of active sessions, queuing inactive ones	ACTIVE_SESS_POOL_ABSOLUTE
QUEUING_MTH: Controlling queues, how inactive sessions enter active session pool	FIFO_TIMEOUT

Resource allocation methods determine how the Resource Manager allocates a particular resource to a resource consumer group or resource plan. You specify values for the following resource allocation methods when creating the resource plan.

There are two ways of specifying the CPU distribution with the MGMT_MTH parameter when you create a resource plan:

- EMPHASIS is the default method for single-level plans. It is also used for multilevel plans that use percentages to specify how CPU is distributed among consumer groups.
- RATIO is for single-level plans that use ratios to specify how CPU is distributed.

PARALLEL_DEGREE_LIMIT_MTH limits the maximum degree of parallelism of any operation. This method can be specified only for resource consumer groups, not subplans. The PARALLEL_DEGREE_LIMIT_ABSOLUTE method is the only possible value, specifying how many processes may be assigned to an operation. If there are multiple plan directives referring to the same subplan or consumer group, the **minimum** of all the possible values is used as the parallel degree limit for that subplan or consumer group.

The ACTIVE_SESS_POOL_MTH parameter limits the number of active sessions. All other sessions are inactive and wait in a queue to be activated. The only value (the only available method) for this parameter is ACTIVE_SESS_POOL_ABSOLUTE, which is its default value.

QUEUING_MTH controls the order in which queued inactive sessions execute. FIFO_TIMEOUT is the default and only method available.

Comparison of EMPHASIS and RATIO

EMPHASIS	RATIO
The value specifies the maximum percentage of CPU resources a consumer group can use.	The value specifies a number that indicates the ratio of CPU resources to be allocated to the consumer group.
You can allocate resources for up to eight different levels.	You can specify values for only one level.
The sum of percentages at any given level must be less than or equal to 100.	You must use integer values, but there is no limit on the sum of values.
Default value is NULL.	Default value is NULL.

The CPU allocation method EMPHASIS determines how much emphasis is given to sessions in different consumer groups in a resource plan. CPU usage is assigned levels from 1 through 8, with Level 1 having the highest priority. Percentages specify how to allocate CPU to each consumer group at each level.

The following rules apply for the EMPHASIS resource allocation method:

- CPU resources are distributed at a given level on the basis of the specified percentages. The percentage of CPU specified for a resource consumer group is a maximum for how much that consumer group can use at a given level.
- Consumer resources that are not used at a given level are made available to consumer groups at the next level. For example, if the consumer groups at Level 1 use only 60% of the available resources, the additional 40% is made available to consumer groups at Level 2.
- The sum of percentages at any given level must be less than or equal to 100.
- Any levels that have no plan directives explicitly specified have a default of 0% for all subplans or consumer groups.
- The EMPHASIS resource allocation method avoids starvation problems, where consumers with lower priorities are not given the opportunity to run.

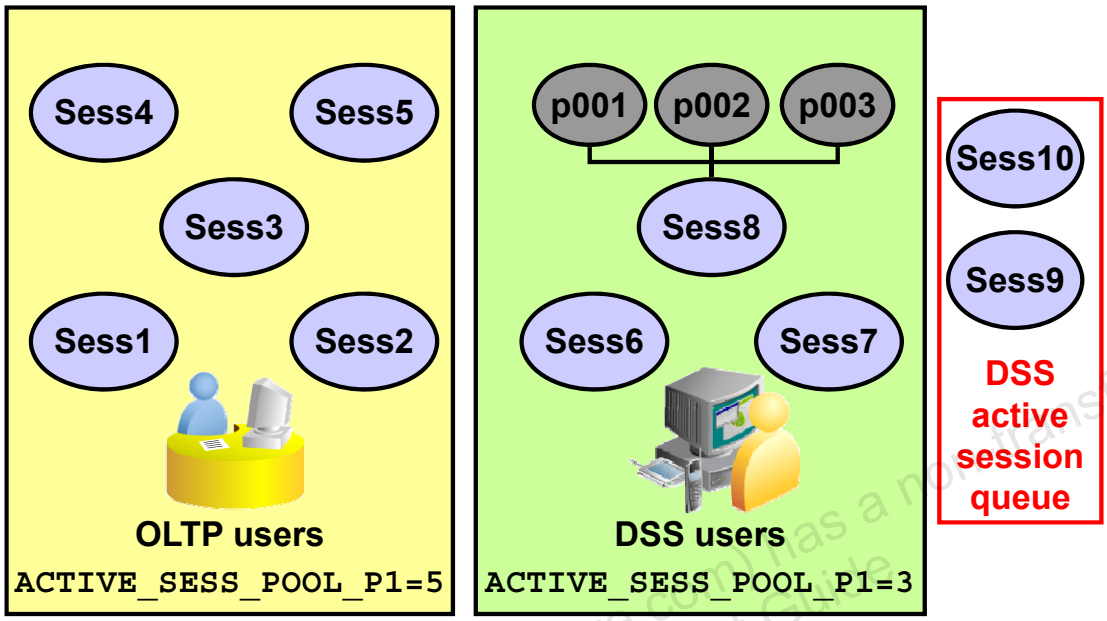
The RATIO policy is a single-level CPU allocation method. Instead of percentages, you specify numbers corresponding to the ratio of CPU you want to give to the consumer group. For example, given three consumer groups OLTP_USERS, DSS_USERS, and BATCH_USERS, you can specify the following ratios:

- OLTP_USERS: 4
- DSS_USERS: 3
- BATCH_USERS: 2
- OTHER: 1

This is similar to saying that OLTP users should get 40% of the resources, DSS users should get 30% of the resources, BATCH users should get 20% of the resources, and all other consumer groups should get 10% of the available resources.

If there are no consumers in the OTHER or DSS_USERS consumer groups currently utilizing CPU resources, the OLTP_USERS consumer group would get two-thirds (4 shares out of 6 shares) of the available resources and the BATCH_USERS consumer group would get the other third (2 shares out of 6). If all groups had sessions running, the division would be based on 10 shares.

Active Session Pool Mechanism



Using the Active Session Pool feature, you can control the maximum number of concurrently active sessions per resource consumer group. With this functionality, a DBA can indirectly control the amount of resources that any resource consumer group uses because resource consumption is proportional to the number of active sessions. Using an active session pool can help to reduce the number of servers taking resources in the system, thus avoiding inefficient paging, swapping, and other resource depletion (such as memory) resulting from attempting to run too many jobs simultaneously.

After the Active Session Pool is filled with active sessions, the Resource Manager queues all subsequent sessions attempting to become active until other active sessions complete or become inactive. An active session is one currently involved in a transaction, query, or parallel operation. Individual parallel slaves are not counted as sessions; the entire parallel operation counts as one active session.

There is only one queue per resource consumer group and the queuing method is first in, first out (FIFO) with a timeout. The queue is implemented as a memory structure and cannot be queried directly.

Specifying Thresholds

Specifying execution time limit:

- Proactive estimation of the execution time for an operation (via cost-based optimizer statistics), default: `UNLIMITED`
- Specifying maximum estimated execution time at the resource consumer group level
- Huge jobs will not be allowed to start if the estimate is longer than `MAX_EST_EXEC_TIME:(ORA-07455)`

Specifying other thresholds:

- Limiting session I/O with `SWITCH_IO_MEGABYTES` (in MB)
- Limiting session I/O requests with `SWITCH_IO_REQS`

Returning to original consumer group with `SWITCH_FOR_CALL` (Default: `FALSE`, consumer group is not restored)

You can define the maximum estimated execution time any operation can take at any given time by setting the resource plan directive's `MAX_EST_EXEC_TIME` parameter.

- When this parameter is set, the Database Resource Manager estimates the time a specific job will take, which is calculated using the statistics from the cost-based optimizer.
- If a resource consumer group has more than one plan directive referring to it, it may have more than one `MAX_EST_EXEC_TIME`. The Database Resource Manager then chooses the most restrictive of all incoming values.
- If the operation's estimate is more than `MAX_EST_EXEC_TIME`, the operation does not start and the `ORA-07455` error is issued. This eliminates any exceptionally large jobs that would utilize too many system resources.
- The `SWITCH_IO_MEGABYTES` directive specifies the amount of I/O (in MB) that a session can issue before an action is taken. The default is `NULL`, which means unlimited.
- The `SWITCH_IO_REQS` directive specifies the number of I/O requests that a session can issue before an action is taken. The default is `NULL`, which means unlimited.
- The `SWITCH_FOR_CALL` directive specifies that if an action is taken because of the `SWITCH_TIME`, `SWITCH_IO_MEGABYTES`, or `SWITCH_IO_REQS` parameters, the consumer group is restored to its original consumer group at the end of the top call. Default is FALSE, which means that the original consumer group is not restored at the end of the top call.

This functionality is mostly beneficial for three-tier applications where the middle-tier server implements session pooling. In this case, the middle tier tends to do one call for an end user and then use the same session for a call for a different end user. Therefore, the boundaries of work are really calls, and the actions of a prior end user should not affect the next end user.

Note: You cannot specify both the `SWITCH_TIME_IN_CALL` and `SWITCH_TIME` parameters within the same directive. The `SWITCH_TIME` parameter is primarily intended for client/server applications, whereas the `SWITCH_TIME_IN_CALL` parameter is for three-tier applications.

Setting Idle Timeouts

```
DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE
 (PLAN => 'DAY_PLAN',
  GROUP_OR_SUBPLAN => 'APPUSER',
  COMMENT => 'Limit Idle Time Example',
  NEW_MAX_IDLE_TIME => 600,
  NEW_MAX_IDLE_BLOCKER_TIME => 300);
```


ORACLE

In Enterprise Manager Cloud Control, use the Idle Time tab to set the maximum idle timeouts for a resource plan. “Max Idle Time (sec)” and “Max Idle Time if Blocking Another Session (sec)” are the respective equivalents of the `NEW_MAX_IDLE_TIME` and `NEW_MAX_IDLE_BLOCKER_TIME` resource directives in the `DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE` procedure. They are both specified in seconds.

`NEW_MAX_IDLE_TIME` specifies the time that a session is neither executing nor waiting for I/O. When the session exceeds the specified limit, the `PMON` process forcibly kills the session and cleans up its state. In addition to limiting the maximum idle time for a session, you can also limit the amount of time that an idle session can block another session. You impose this limit by setting the `NEW_MAX_IDLE_BLOCKER_TIME` resource directive to the number of seconds to allow a session to be idle while blocking another session. You can also specify a value of `UNLIMITED` to indicate that no maximum time has been set. The default is `NULL`, which means unlimited. These settings give you a more granular control than profiles, whose single value cannot distinguish between blocking and nonblocking sessions.

In the slide example, the `PMON` process kills sessions that are idle for longer than 600 seconds. The `PMON` process also kills sessions that are idle for more than 300 seconds and are blocking other sessions. `PMON` checks these limits once every minute and if it finds a session that has exceeded one of the limits, it forcibly kills the session and cleans up all its resources.

Limiting CPU Utilization at the Database Level

- Database consolidation requirements:
  - Applications isolated from each other
  - Consistent performance
- CPU directives can be used to:
  - Specify a minimum CPU allocation for each application
  - Designate how unused allocations should be redistributed
  - Specify the `MAX_UTILIZATION_LIMIT` attribute to impose an absolute upper limit on CPU utilization (which overrides any redistribution of CPU within a plan)
  - Good candidate: Auto-maintenance tasks

For concurrent database sessions: Database consolidation requires that applications are isolated from each other. If one application experiences an increase in workload, that increase should not affect other applications. In addition, the performance of each application should be consistent.

Fixed Policy CPU Resource Management

The `MAX_UTILIZATION_LIMIT` attribute of resource plan directives enables you to impose an absolute upper limit on CPU utilization for a resource consumer group. This absolute limit overrides any redistribution of CPU within a plan.

Note: Good candidate applications for database consolidation are automated maintenance tasks because currently these applications can take up to 100% of the server CPU resources. You can set a maximum limit for each auto-task consumer group.

Limiting CPU Utilization at the Database Level

Specify minimum and maximum CPU utilization limits.

DB Consolidation Plan #1

	CPU Allocation	Maximum Utilization Limit
App 1	50%	60%
App 2	20%	30%
App 3	20%	30%
App 4	10%	20%

Specify maximum CPU utilization limits only.

DB Consolidation Plan #2

	CPU Allocation	Maximum Utilization Limit
App 1	null	50%
App 2	null	20%
App 3	null	20%
App 4	null	10%

```
EXEC DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE( -
     plan                    => 'db_consolidation_plan',
     group_or_subplan        => 'App_1',
     mgmt_p1                 => 50,
     max_utilization_limit => 60);
```


The `MAX_UTILIZATION_LIMIT` directive limits the CPU consumption of an application. You can set minimum and maximum boundaries, as shown in the slide.

The PL/SQL example in the slide specifies a minimum value percentage (50%) for the CPU allocation resource at level 1 for the `APP_1` consumer group. This example also specifies an absolute maximum CPU utilization percentage (60%) permitted for that same consumer group. The example uses the `DB_CONSOLIDATION_PLAN` plan.

Similar commands can be executed for each consumer group shown in the sample tables.

Limiting CPU Utilization at the Server Level: Instance Caging

- Managing CPU allocations on a multi-CPU server with multiple database instances
- Enabling instance caging:
  - Enable any CPU resource plan.

```
ALTER SYSTEM SET resource_manager_plan = 'default_plan';
```

  - Specify the maximum number of CPUs that the instance can use at any time.

```
ALTER SYSTEM SET cpu_count=4;
```

- Two approaches:
  - Over-provisioning: The sum of the CPU limit for each instance exceeds the actual number of CPUs.
  - Partitioning: The sum of the CPU limit for each instance equals the actual number of CPUs.

ORACLE

Because many test, development, and small production databases are unable to fully utilize the servers that they are on, *server consolidation* provides a possible alternative. With server consolidation, resources are more fully utilized by running multiple database instances on the server. However, this may bring about CPU contention and an adverse impact due to workload surges on one instance.

Instance caging is a method that uses the `CPU_COUNT` initialization parameter to limit the number of CPUs that an instance can use. In addition, the Resource Manager is employed to allocate the CPUs for the database sessions based on the instance resource plan.

Configure instance caging in two steps, by enabling:

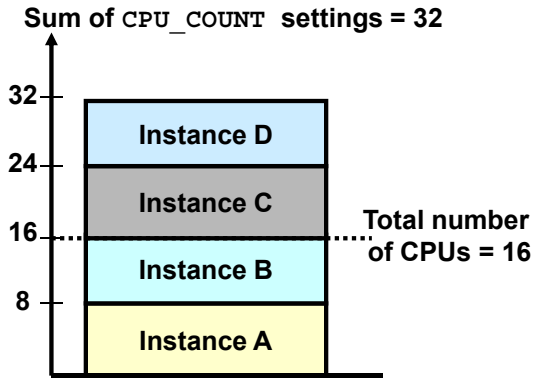
- The Resource Manager, which limits the amount of CPU that the database instance consumes
- The `CPU_COUNT` parameter, which specifies the maximum (the limit), not actual amount of CPU that the database instance can use at any time

By default, the CPU Resource Manager assumes that the database instance can use all CPUs on a server. To enable instance caging, any resource plan with CPU directives can be used.

Instance Caging Examples

Over-provisioning approach:

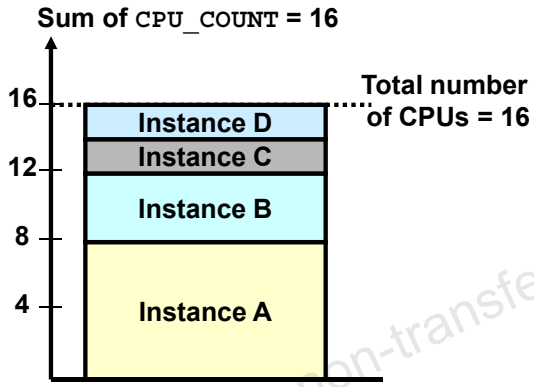
One database instance can still impact the others.



With all four instances active, one instance can get 8 / (8 + 8 + 8 + 8) = 25% of CPU.

Partitioning approach:

One database instance cannot impact the others.



Each instance has a dedicated number of CPUs.

Over-provisioning approach: This approach is appropriate for noncritical databases and low-load, noncritical production systems. Although the instances impact each other's performance, at any given time, one or more of the instances may be idle or experiencing a low load.

Although the database instances can impact each other's performance, instance caging limits their impact and helps to provide predictable performance. In the example on the left, where all four instances have `CPU_COUNT` set to `8`, the maximum percentage of CPU that a database instance can consume at any point in time is its own limit divided by the sum of the limits for all active databases. In this example, one instance will be able to consume 8 / (8 + 8 + 8 + 8) = 25% of the CPU. If only two instances are active, one instance will be able to consume 8 / (8 + 8) = 50% of the CPU.

Partitioning approach: This approach is appropriate for critical product systems. It prevents the instances from interfering with each other and provides predictable performance.

Instance caging can partition the CPU resources by ensuring that the sum of all CPU limits does not exceed the total number of CPUs. In the example on the right, if four database instances share a 16-CPU server, their limits can be set to 8, 4, 2, and 2. By dedicating CPU resources to a database instance, partitioning provides two advantages:

- One database instance's CPU load cannot affect another's.
- Each database instance's CPU resources is fixed, leading to more predictable performance.

Monitoring Instance Caging

View value of the CPU_COUNT parameter:

```
SELECT value FROM v$parameter WHERE name = 'cpu_count'
   AND (isdefault = 'FALSE' OR ismodified != 'FALSE');
```

Determine the Resource Manager status:

```
SELECT name FROM v$rsrc_plan
WHERE is_top_plan = 'TRUE' AND cpu_managed = 'ON';
```

Manage throttling:

```
SELECT begin_time, consumer_group_name,
   cpu_consumed_time, cpu_wait_time
FROM v$rsrcmgrmetric_history
ORDER BY begin_time;
```

```
SELECT name, consumed_cpu_time, cpu_wait_time
FROM v$rsrc_consumer_group;
```


- If the CPU_COUNT parameter is not set, no value is returned by the first query.
- If no rows are returned by the second query in the slide, the Resource Manager is not managing the CPU. If a row is returned, it indicates the active plan.

Instance caging limits the CPU consumption of the foreground processes by throttling them. A foreground process is throttled when it is waiting on the “resmgr:cpu quantum” wait event.

You can monitor the amount of throttling in two ways:

- The V$RSRCMGRMETRIC_HISTORY view shows the amount of CPU consumption (CPU_CONSUMED_TIME) and throttling (CPU_WAIT_TIME) for each minute in the past hour. Values are displayed in milliseconds.
- The V$RSRC_CONSUMER_GROUP view shows the amount of CPU consumption (CPU_CONSUMED_TIME) and throttling (CPU_WAIT_TIME) since CPU Resource Management was enabled. The time is displayed in milliseconds.

Note: For case studies, see the Oracle White Paper titled *Database Instance Caging: A Simple Approach to Server Consolidation.*

Runaway Queries and Resource Manager

- Parameters used to trigger consumer group switching:
  - SWITCH_IO_LOGICAL
  - SWITCH_ELAPSED_TIME
- Meta consumer group called LOG_ONLY
- Columns in V$SQL_MONITOR:
  - RM_LAST_ACTION
  - RM_LAST_ACTION_REASON
  - RM_LAST_ACTION_TIME
  - RM_CONSUMER_GROUP
- V$RSRCMGRMETRIC and V$RSRCMGRMETRIC_HISTORY always populated

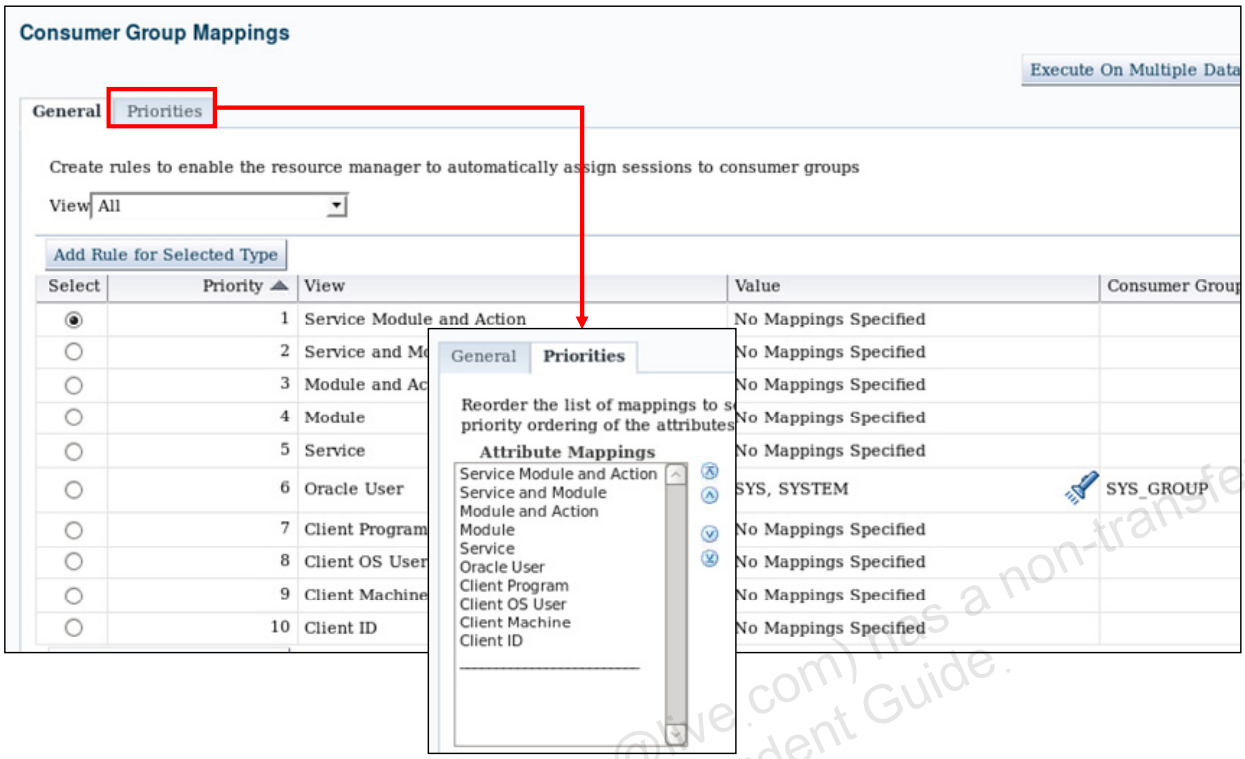
To better track runaway queries, Resource Manager included the following:

- Directive parameters:
  - SWITCH_IO_LOGICAL: Number of logical I/Os that will trigger the action specified by SWITCH_GROUP
  - SWITCH_ELAPSED_TIME: Elapsed time that will trigger the action specified by SWITCH_GROUP
- LOG_ONLY meta-consumer group: This can be used as the argument for the SWITCH_GROUP parameter to log the runaway query without changing its consumer group or taking other action.
- Resource Manager integrates the runaway query information with SQL Monitor. To retain important results for Resource Manager, it pins up to five SQL statements per consumer group. SQL Monitor does not purge these SQL executions until they are unpinned. V$SQL_MONITOR columns:
  - RM_LAST_ACTION: The most recent action that was taken on this SQL operation by Resource Manager. Its value is one of the following: CANCEL_SQL, KILL_SESSION, LOG_ONLY, SWITCH TO <CG NAME>

- RM_LAST_ACTION_REASON: The reason for the most recent action that was taken on this SQL operation by Resource Manager. Its value is one of the following: SWITCH_CPU_TIME, SWITCH_IO_REQS, SWITCH_IO_MBS, SWITCH_ELAPSED_TIME, SWITCH_IO_LOGICAL
- RM_LAST_ACTION_TIME: The time of the most recent action that was taken on this SQL operation by Resource Manager
- RM_CONSUMER_GROUP: The current consumer group for this SQL operation

Note: V$RSRCMGRMETRIC and V$RSRCMGRMETRIC_HISTORY will always produce a row every minute regardless of whether there is a Resource Manager plan set.

Resource Consumer Group Mapping



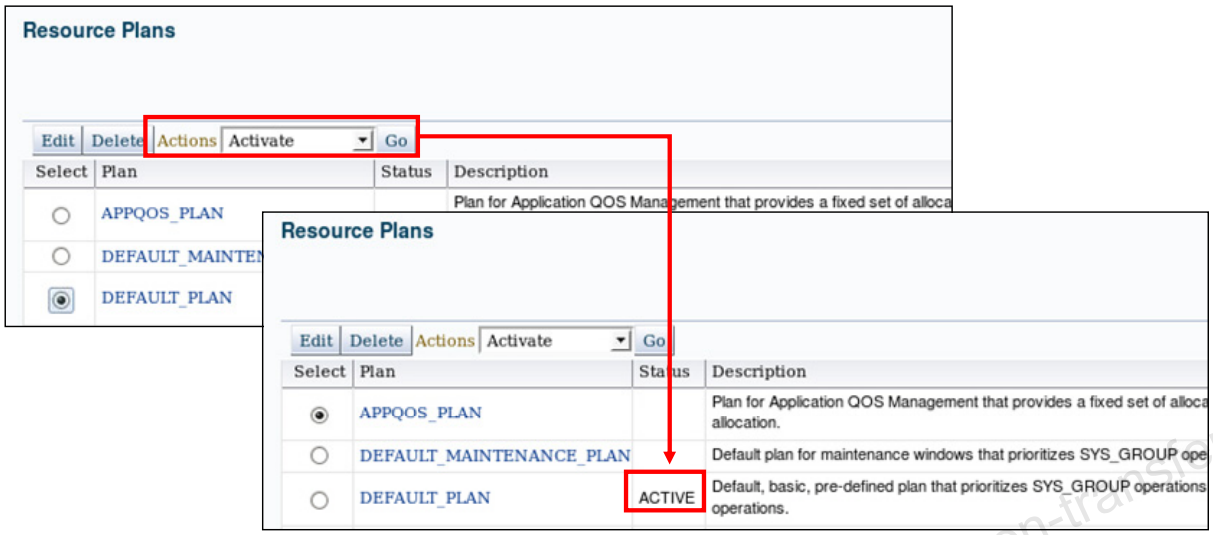
You can configure the Database Resource Manager to automatically assign consumer groups to sessions by providing mappings between session attributes and consumer groups. Further, you can prioritize the mappings so as to indicate which mapping has precedence in case of conflicts. There are two types of session attributes: login attributes and runtime attributes. The login attributes are meaningful only at session login time, when the Database Resource Manager determines the initial consumer group of the session. In contrast, a session that has already logged in can later be reassigned to another consumer group on the basis of its runtime attributes.

Select Consumer Group Mappings on the Getting Started with Database Resource Manager page of Enterprise Manager Cloud Control. For each of the attributes, set up a mapping that consists of a way to identify a session (for example, username), and a consumer group. Add or remove rows for each of the resource consumer group categories, as required, and enter text identifying the user, client, module, or service in the corresponding group. You can establish a priority ordering between conflicting mappings of the attributes by using the Priorities tab. You can set the priority from the most important to the least important by using the navigational arrows. The mappings at the top of the list have the highest priority.

Example to give the Client OS User a higher priority than the Client Program:

```
BEGIN
dbms_resource_manager.clear_pending_area();
dbms_resource_manager.create_pending_area();
dbms_resource_manager.set_consumer_group_mapping(
    dbms_resource_manager.oracle_user,
    'SCOTT',
    'LOW_GROUP'
);
dbms_resource_manager.set_consumer_group_mapping_pri(
    EXPLICIT => 1,  SERVICE_MODULE_ACTION => 2,
    SERVICE_MODULE => 3,
    MODULE_NAME_ACTION => 4,
    MODULE_NAME => 5,
    SERVICE_NAME => 6,
    ORACLE_USER => 7,
    CLIENT_OS_USER => 8,
    CLIENT_PROGRAM => 9,
    CLIENT_MACHINE => 10
);
dbms_resource_manager.submit_pending_area();
END;
```

Activating a Resource Plan




```
SQL> show parameter resource_manager_plan

NAME                      TYPE        VALUE
------------------------- ----------- -------------
resource_manager_plan     string      DEFAULT_PLAN
```


The plan for an instance is defined using the `RESOURCE_MANAGER_PLAN` database initialization parameter. This parameter specifies the top plan to be used for this instance. If no plan is specified, the Resource Manager is not activated for the instance.

You can activate, deactivate, or change the current top plan by using an `ALTER SYSTEM` statement. When a resource plan is changed using this command, the change takes effect instantly.

If the parameter is set in a parameter file, and the plan specified is not defined in the database, the database cannot be opened with that parameter file. The following error is returned:

```
ORA-07452: specified resource manager plan does not exist in the
data dictionary
```

If this error is encountered, the parameter must be modified to show a correct value before the instance can be restarted.

You can use the Resource Plans page of Enterprise Manager Cloud Control to manage resource plans. To activate a plan, select the plan you want to make active, choose Activate from the Actions drop-down list, and then click Go. The plan you selected is then made the current top plan for the instance.

Database Resource Manager Information

View Name	Information
DBA_RSRC_PLANS	Plans and status
DBA_RSRC_PLAN_DIRECTIVES	Plan directives
DBA_RSRC_CONSUMER_GROUPS	Consumer groups
DBA_RSRC_CONSUMER_GROUP_PRIVS	Users/roles
DBA_RSRC_GROUP_MAPPINGS	Consumer group mapping
DBA_RSRC_MAPPING_PRIORITY	Mapping priority
DBA_USERS	Column INITIAL_RSRC_CONSUMER_GROUP
DBA_RSRC_MANAGER_SYSTEM_PRIVS	Users/roles

Several data dictionary views are available to check the resource plans, consumer groups, and plan directives that are declared in the instance. This section discusses some useful information that can be obtained from these views. For more detailed information about the contents of each of these views, refer to the *Oracle Database Reference* manual.

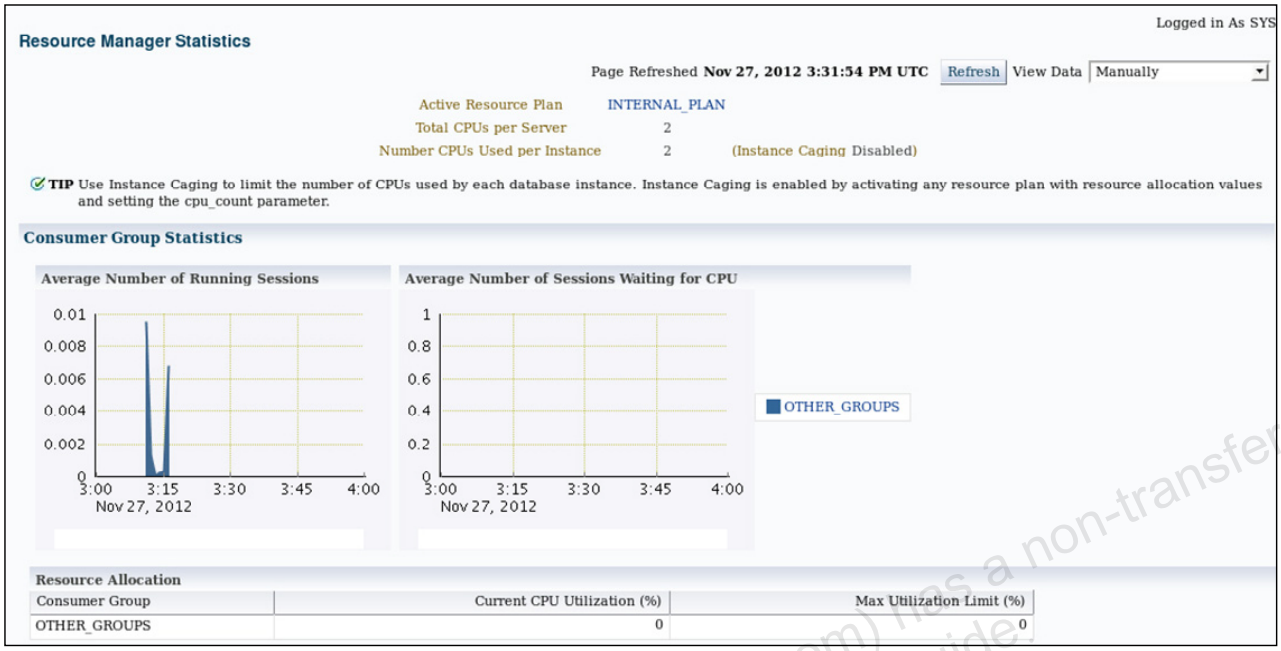
Use the following query to obtain information about resource plans defined in the database:

```
SQL> SELECT plan, num_plan_directives, status, mandatory
  2  FROM dba_rsrc_plans;
PLAN             NUM_PLAN_DIRECTIVES STATUS    MAN
---------------- ------------------- --------- ---
DEFAULT_PLAN                       3 ACTIVE    NO
INTERNAL_QUIESCE                   2 ACTIVE    YES
INTERNAL_PLAN                      1 ACTIVE    YES
BUGDB_PLAN                         4 ACTIVE    NO
MAILDB_PLAN                        3 ACTIVE    NO
MYDB_PLAN                          3 ACTIVE    NO
```

A status of ACTIVE indicates that the plan has been submitted and can be used, whereas, a status of PENDING shows that the plan has been created, but is still in the pending area.

If the MANDATORY column is assigned a value of YES, the plan cannot be deleted.

Viewing Resource Manager Statistics



You can access Resource Manager statistics by selecting Performance Statistics on the Getting Started with Database Resource Manager page in Enterprise Manager Cloud Control. Information is provided so that you can monitor the currently enabled resource plan.

Monitoring the Resource Manager

- V$SESSION: Contains the RESOURCE_CONSUMER_GROUP column that shows the current group for a session
- V$RSRC_PLAN: A view that shows the active resource plan
- V$RSRC_CONSUMER_GROUP: A view that contains statistics for all active groups

CPU Utilization

There are at least three different views in the system that can provide you with information about the CPU utilization inside the Oracle database:

- V$RSRC_CONSUMER_GROUP shows CPU utilization statistics on a per consumer group basis, if you are running the Oracle Database Resource Manager. This view displays data related to currently active resource consumer groups.
- V$SYSSTAT shows the Oracle database CPU usage for all sessions. The statistic “CPU used by this session” shows the aggregate CPU used by all sessions.
- V$SESSTAT shows the Oracle database CPU usage per session. You can use this view to determine which particular session is using the most CPU.

The V$RSRC_CONSUMER_GROUP view contains the following columns:

- **NAME:** Name of the consumer group
- **ACTIVE_SESSIONS:** Number of currently active sessions in this consumer group
- **EXECUTION_WAITERS:** Number of active sessions waiting for a time slice
- **REQUESTS:** Cumulative number of requests executed in this consumer group
- **CPU_WAIT_TIME:** Cumulative amount of time that sessions waited for CPU
- **CONSUMED_CPU_TIME:** Cumulative amount of CPU time consumed by all sessions

There is no view that shows the Active Session Pool queue directly, but you can get some information from:

- **V$SESSION:** The `CURRENT_QUEUE_DURATION` column shows how long a session has been queued, or 0 (zero) if the session is not currently queued.
- **V$RSRC_CONSUMER_GROUP:** The `QUEUE_LENGTH` column shows the number of sessions currently queued per consumer group.

Quiz

Select the statements that are true about the Resource Manager and its functionality:

a. You can set threshold values only for execution time, not for session I/O.
b. You can limit CPU utilization at the database level to isolate applications for each other.
c. On a multi-CPU server with multiples database instances, you can limit each server's CPU utilization by enabling instance caging.
d. When the `SWITCH_TIME`, `SWITCH_IO_MEGABYTES`, or `SWITCH_IO_REQS` parameters cause a switch in consumer groups, you can never return to the original consumer groups.

Answer: b, c

Summary

In this lesson, you should have learned how to do the following:

- Configure the Database Resource Manager
- Access and create resource plans
- Create consumer groups
- Specify directives for allocating resources to consumer groups
- Map consumer groups to plans
- Activate a resource plan
- Monitor the Resource Manager

Practice A

A-1: Managing Resources