Resources Allocation

Objectives

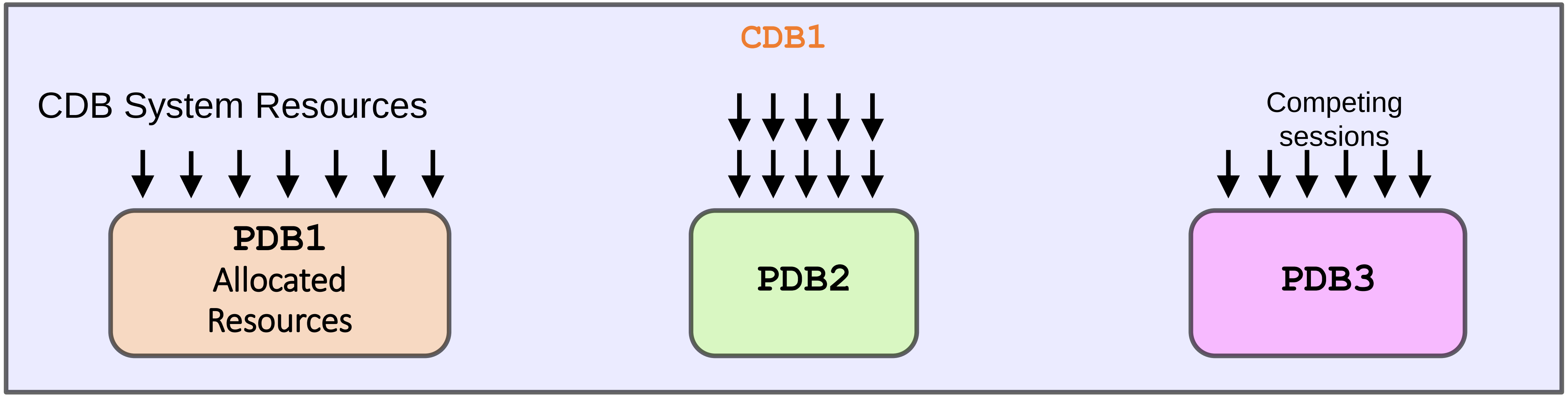
- After completing this lesson, you should be able to:
  - Manage resource allocation between PDBs and within a PDB
  - Control PDB IO rate limit
  - Enable parallel statement queuing at PDB level
  - Avoid excessive session PGA
  - Manage PDB performance profiles

Allocating Resources in the CDB

- Choose a strategy:
  - Allow all PDBs to use all the resources.
    - Gives maximum flexibility for each PDB
    - Allows any PDB to consume all available resources
  - Assign a minimum allocation to each PDB.
    - Ensures all PDBs get a specific share of the resources
    - Allows any PDB to consume any unused resources
  - Assign a maximum allocation to each PDB.
    - Prevents a PDB from taking more than the maximum value assigned
    - May result in unused capacity

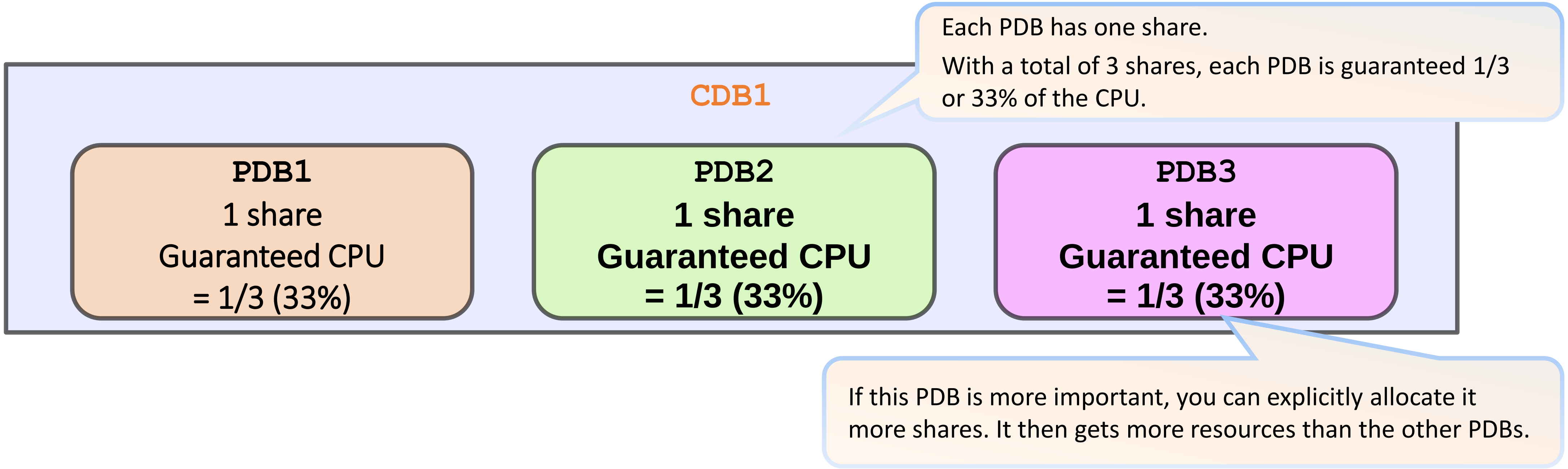
Resource Manager and Pluggable Databases

- In a CDB, the Resource Manager manages resources:
  - Between PDBs
  - Within each PDB



Managing Resources Between PDBs

- PDBs compete for resources: CPU, Exadata I/O, and parallel servers
  - System shares are used to allocate resources for each PDB.
  - Limits are used to cap resource utilization of each PDB.
- When a new PDB is plugged in, the CDB DBA can specify a default or an explicit allocation.



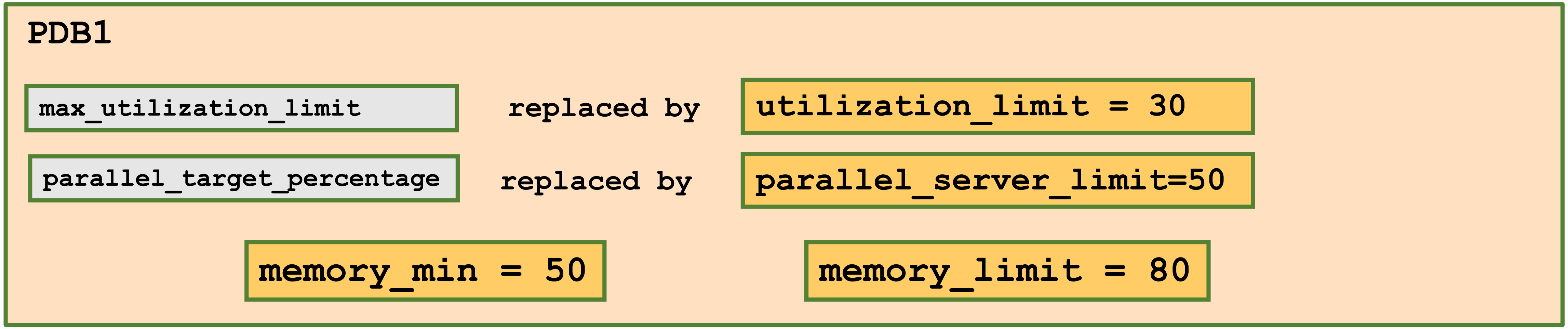
Each PDB has one share.

With a total of 3 shares, each PDB is guaranteed 1/3 or 33% of the CPU.

If this PDB is more important, you can explicitly allocate it more shares. It then gets more resources than the other PDBs.

CDB Resource Plan Basics: Limits

- Four limits can be defined for each PDB:
  - Utilization limit for CPU, Exadata I/Os, and parallel servers
  - Parallel server limit to override the utilization limit
  - `Memory_min`
  - `Memory_limit`
- You can change default values.



PDB1

`max_utilization_limit` replaced by `utilization_limit = 30`

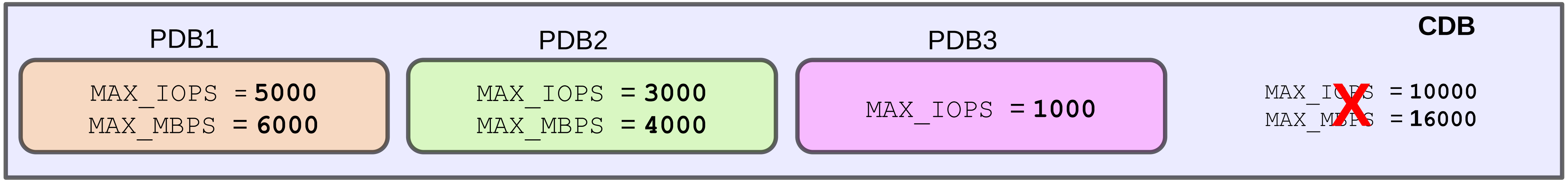
`parallel_target_percentage` replaced by `parallel_server_limit=50`

`memory_min = 50`

`memory_limit = 80`

PDB IO Rate Limit

- `MAX_IOPS`: Number of IOs issued per second
- `MAX_MBPS`: MB of IO issued per second
- Set to 0 by default *(no value at the CDB root level)* → no limit
- Stored in the PDB dictionary
- Migrated with the PDB on an unplug or a plug into a new CDB



PDB1

MAX_IOPS = 5000
MAX_MBPS = 6000

PDB2

MAX_IOPS = 3000
MAX_MBPS = 4000

PDB3

MAX_IOPS = 1000

CDB

MAX_IOPS = 10000
MAX_MBPS = 16000

```
SQL> ALTER SESSION SET CONTAINER = PDB1;
SQL> ALTER SYSTEM SET MAX_IOPS = 5000 SCOPE=SPFILE;
SQL> ALTER SYSTEM SET MAX_MBPS = 6000 SCOPE=SPFILE;
```

DBA_HIST_RSRC_PDB_METRIC

IO_REQUESTS, IO_MEGABYTES,
IOPS, IOMBPS,
AVG_IO_THROTTLE

CDB Resource Plan: Full Example

Specifies allocation of CPU, Exadata I/O, parallel servers

Limits usage of CPU, Exadata I/O

Limits usage of parallel servers

PDB/Directive Name	Shares	Utilization Limit	Parallel Server Limit	Memory Limit	Memory Minimum
(Default Allocation)	(1)	(100%)	(100%)	(100%)	
(Autotask Allocation)	(-1)	(90%)	(100%)	(100%)	
PDB1	1	30%	50%	50%	30%
PDB2	1	30%	80%	40%	10%
PDB3	1	30%	30%	30%	10%
PDB4	2	100%	100%	100%	50%

Sets minimum of memory

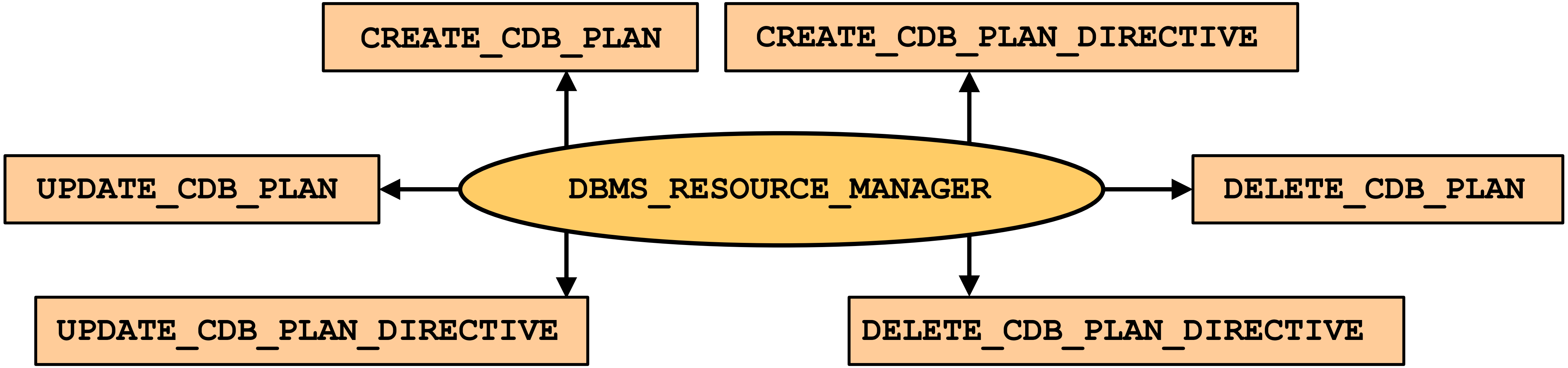
Limits usage of memory

PDB1 is:

- Guaranteed 1/5 (20%) of CPU and Exadata disk bandwidth
- Limited to 30% of CPU and Exadata disk bandwidth
- Limited to 50% of the parallel servers

Maintaining a CDB Resource Plan

```
SQL> CONNECT / AS SYSDBA
SQL> EXEC DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
```



CREATE_CDB_PLAN
CREATE_CDB_PLAN_DIRECTIVE
UPDATE_CDB_PLAN
DBMS_RESOURCE_MANAGER
DELETE_CDB_PLAN
UPDATE_CDB_PLAN_DIRECTIVE
DELETE_CDB_PLAN_DIRECTIVE

```
SQL> EXEC DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA()
SQL> EXEC DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA()
```

Managing Resources Within a PDB

- In a non-CDB database, workloads within a database are managed with resource plans.
- In a PDB, workloads are also managed with resource plans, also called PDB resource plans.
- The functionality is similar except for the following differences:

V$RSRC_PLAN

Non-CDB Database	PDB Database
Multilevel resource plans	Single-level resource plans only
Up to 32 consumer groups	Up to eight consumer groups
Subplans	No subplans

Putting It Together

- How do CDB and PDB resource plans work together?
  - Resources allocated to a PDB, based on CDB resource plan
  - Resource allocated to a consumer group based on the PDB resource plan

CDB Plan

PDB	Shares	Utilization Limit
PDB1	1	50%
PDB2	1	50%
PDB3	2	100%

PDB3 Plan

Consumer Group	Shares	Utilization Limit
OLTP	3	100%
REPORTS	1	50%
OTHER	1	50%

- What does this mean for PDB3 Reports?
  - Guaranteed 50% (2/4) x 20% (1/5) = 10% of the resources
  - Limited to 100% x 50% = 50% of the resources

Considerations

- Setting a PDB resource plan is optional. If not specified, all sessions within the PDB are treated equally.
- A non-CDB is plugged into a CDB with a plan:
  - The plan runs exactly the same on the PDB if:
    - Consumer groups <= 8
    - There are no subplans
    - All allocations are on level 1
  - The plan is converted.
    - The original plan is stored in the dictionary with the LEGACY status.

19c

PDB-Level Parallel Statement Queuing

- Possible issues of parallel statements queuing in a PDB:

```
PARALLEL_DEGREE_POLICY = AUTO | ADAPTIVE
```

  - Not sufficient parallel servers available
  - Parallel statements queued for a long time
- A PDB DBA can make parallel statement queuing work just as it does in a non-CDB.
  - Disable parallel statement queuing at CDB level: PARALLEL_SERVERS_TARGET = 0.
  - Set the PARALLEL_SERVERS_TARGET initialization parameter for individual PDBs.
  - Kill a runaway SQL operation:

```
SQL> ALTER SYSTEM CANCEL SQL '272,31460';
```

  - Dequeue a parallel statement:

```
SQL> EXEC dbms_resource_manager.dequeue_parallel_statement()
```

  - Define the action when dequeuing: PQ_TIMEOUT_ACTION plan directive

PDB-Level Parallel Statement Queuing: CPU_COUNT

19c

- If CPU_COUNT is set at the PDB level, the maximum DOP generated by AutoDOP queries are the PDB's CPU_COUNT.
- Similarly, the default values for PARALLEL_MAX_SERVERS and PARALLEL_SERVERS_TARGET are computed based on the PDB's CPU_COUNT.

Session PGA Limit

- For security purposes:
  - Avoid excessive usage of PGA memory
  - Set the PGA limit that a session can use before it hits an error

`DAY` plan

Consumer Group	Session_PGA_limit
OLTP	350
REPORTS	100
OTHER	50

`NIGHT` plan

Consumer Group	Session_PGA_limit
OLTP	50
REPORTS	400
OTHER	50

V$RSRC_CONSUMER_GROUP

…

PGA_LIMIT_SESSIONS_KILLED

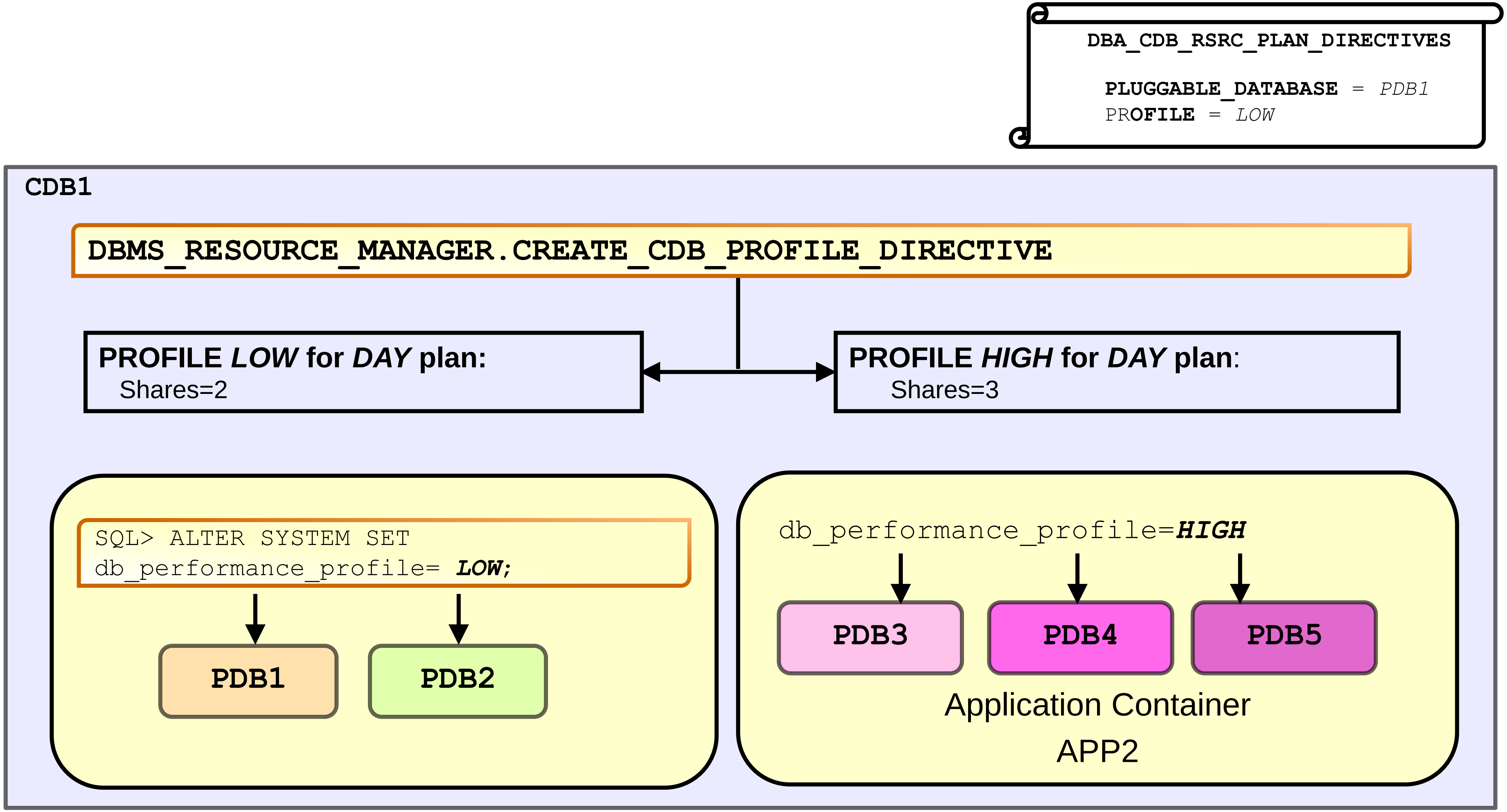
V$RSRC_CONS_GROUP_HISTORY

…

PGA_LIMIT_SESSIONS_KILLED

The number of times that sessions in a consumer group hit the session PGA limit

Performance Profiles


DBA_CDB_RSRC_PLAN_DIRECTIVES
PLUGGABLE_DATABASE = PDB1
PROFILE = LOW
CDB1
DBMS_RESOURCE_MANAGER.CREATE_CDB_PROFILE_DIRECTIVE
PROFILE LOW for DAY plan:
Shares=2
PROFILE HIGH for DAY plan:
Shares=3
SQL> ALTER SYSTEM SET
db_performance_profile= LOW;
PDB1
PDB2
db_performance_profile=HIGH
PDB3
PDB4
PDB5
Application Container
APP2

Summary

- In this lesson, you should have learned how to:
  - Manage resource allocation between PDBs and within a PDB
  - Control PDB IO rate limit
  - Enable parallel statement queuing at PDB level
  - Avoid excessive session PGA
  - Manage PDB performance profiles

Practice 11: Overview

- 11-1: Managing PDB performance profiles
- 11-2: Managing resource allocation between PDBs
- 11-3: Avoiding excessive session PGA memory usage in PDBs