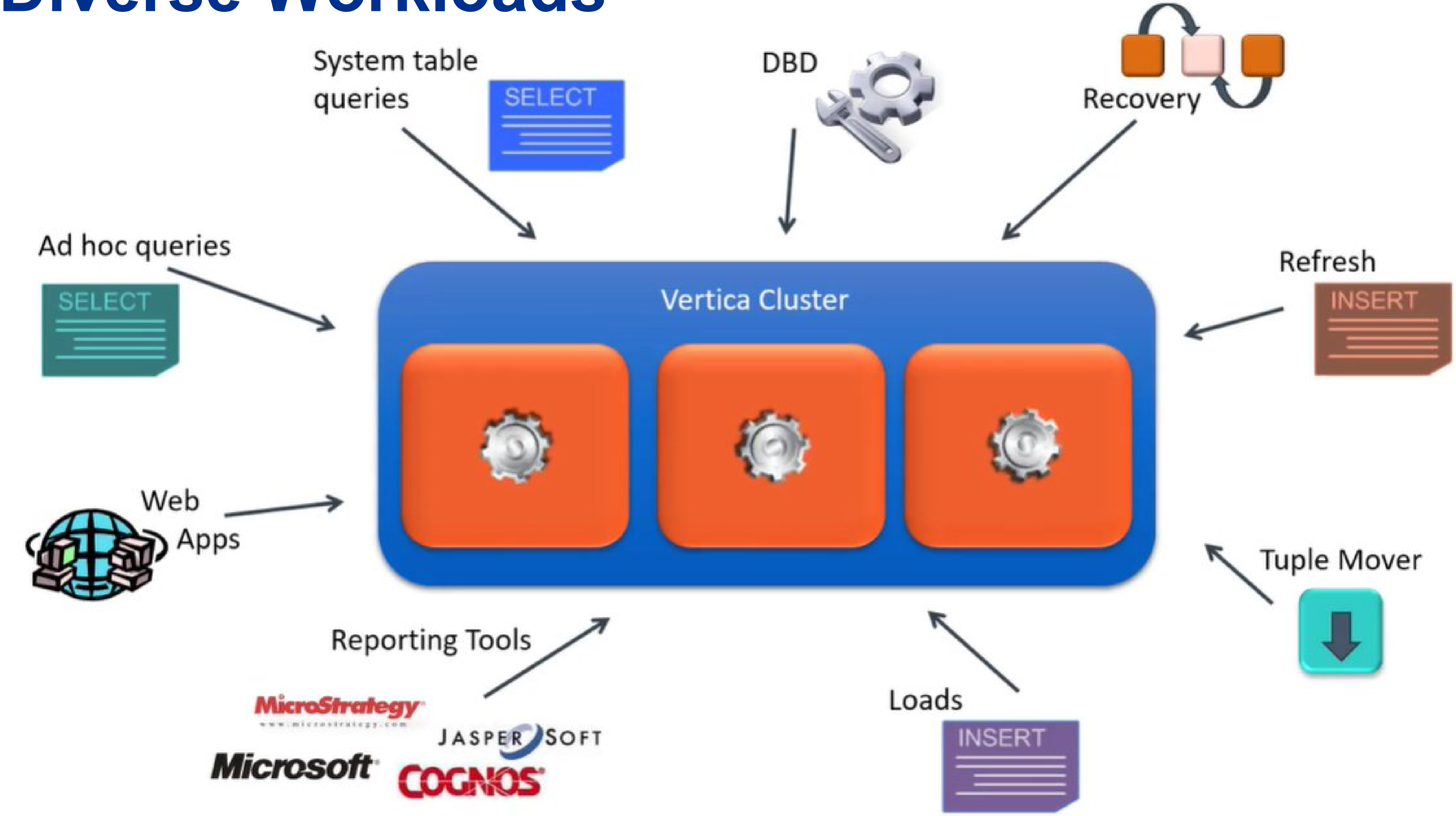
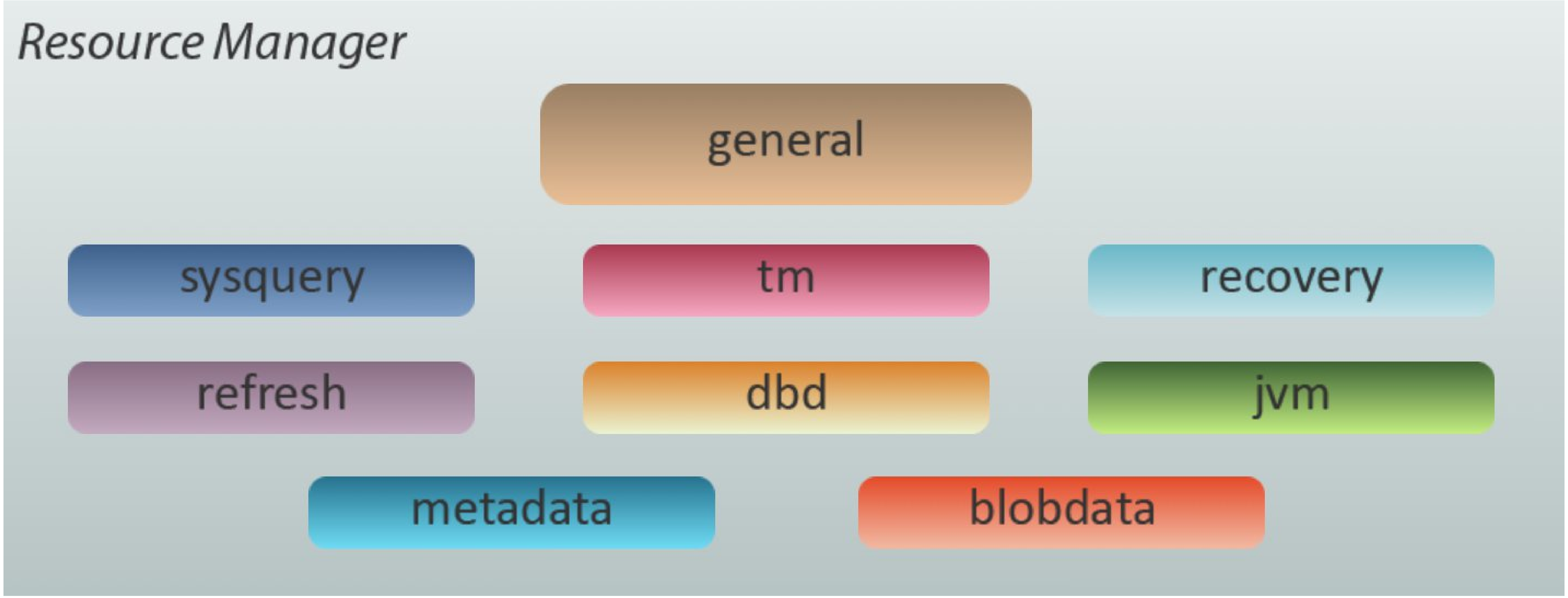
VERTICA
by opentext™

Workload Management

Diverse Workloads



Built in resource pools



Resource Manager

general

sysquery

tm

recovery

refresh

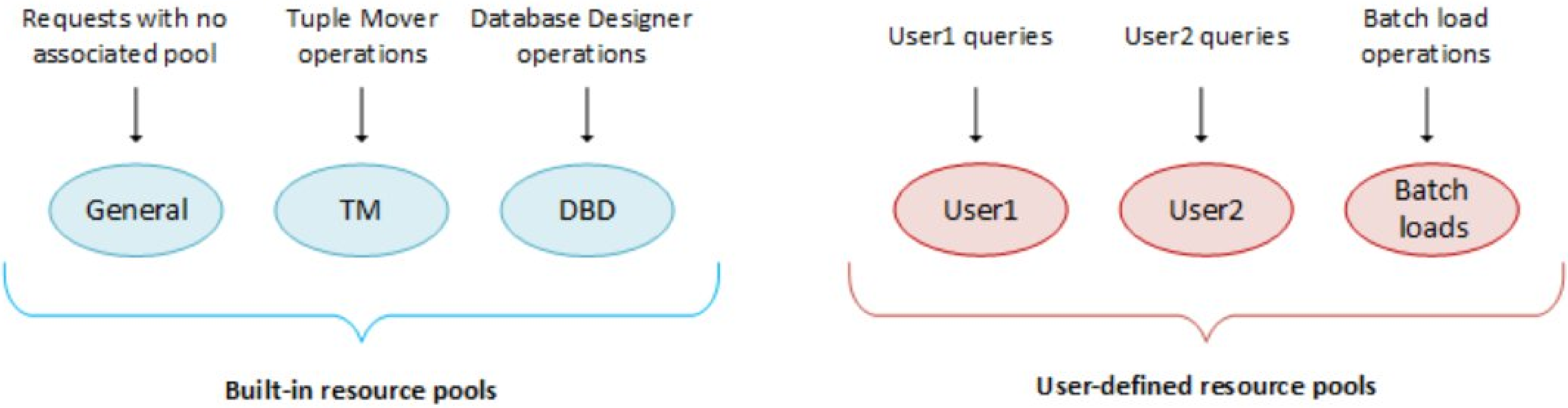
dbd

jvm

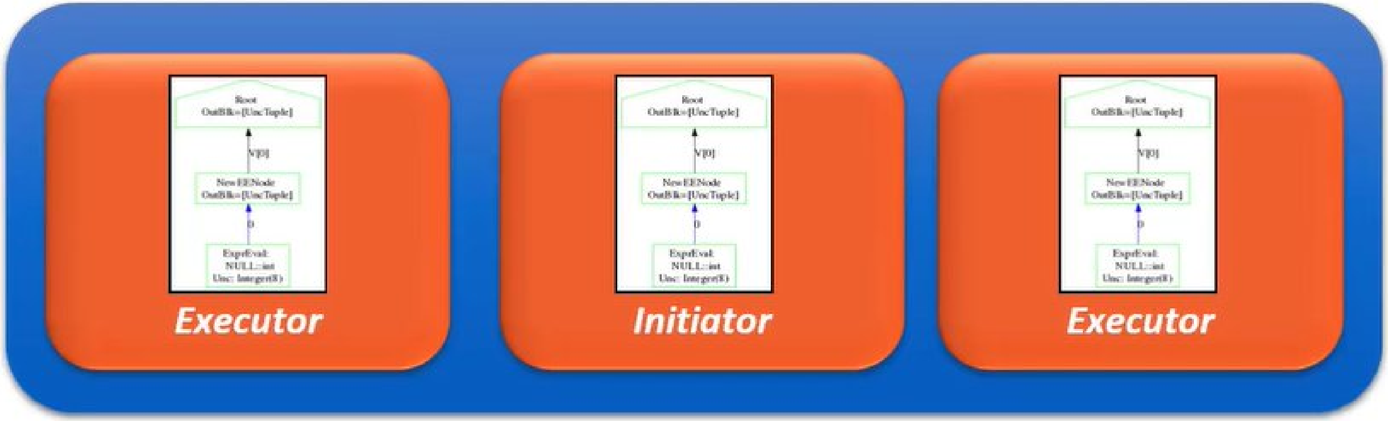
metadata

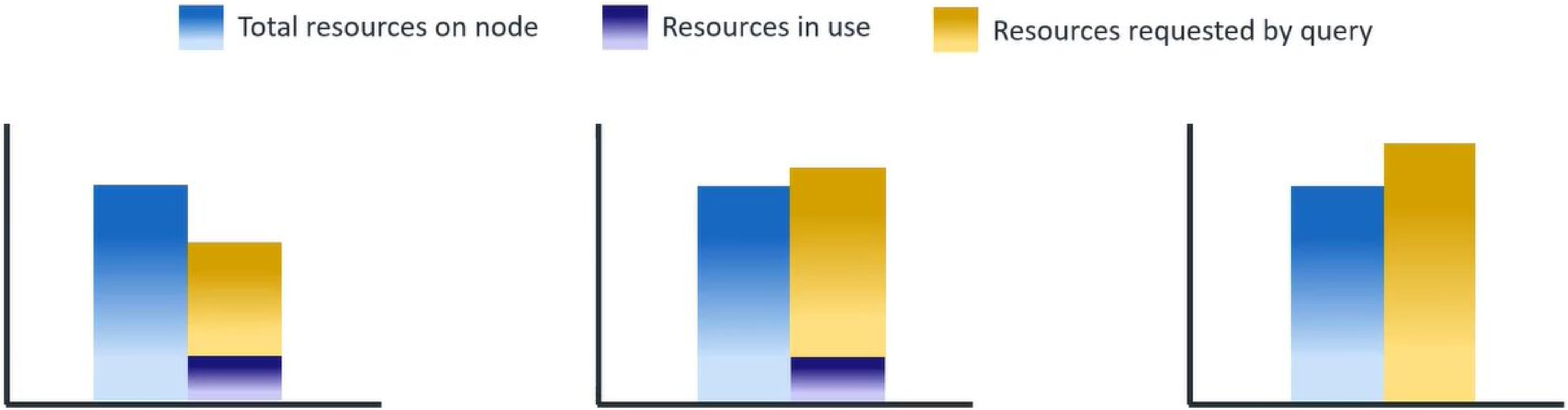
blobdata

Resource Pool Architecture



Query execution and Resource Manager



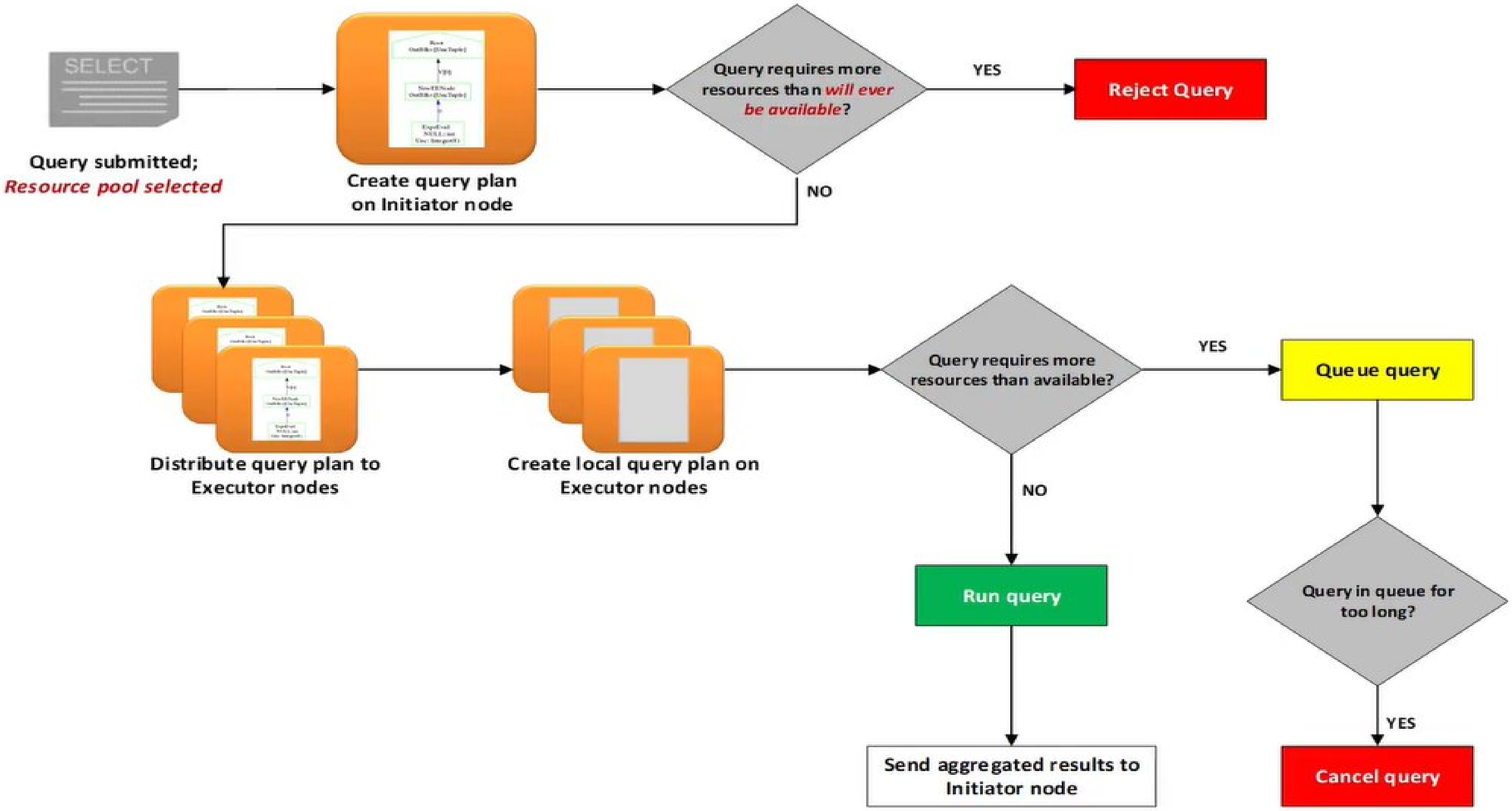


Query **runs immediately**

Query is **queued**

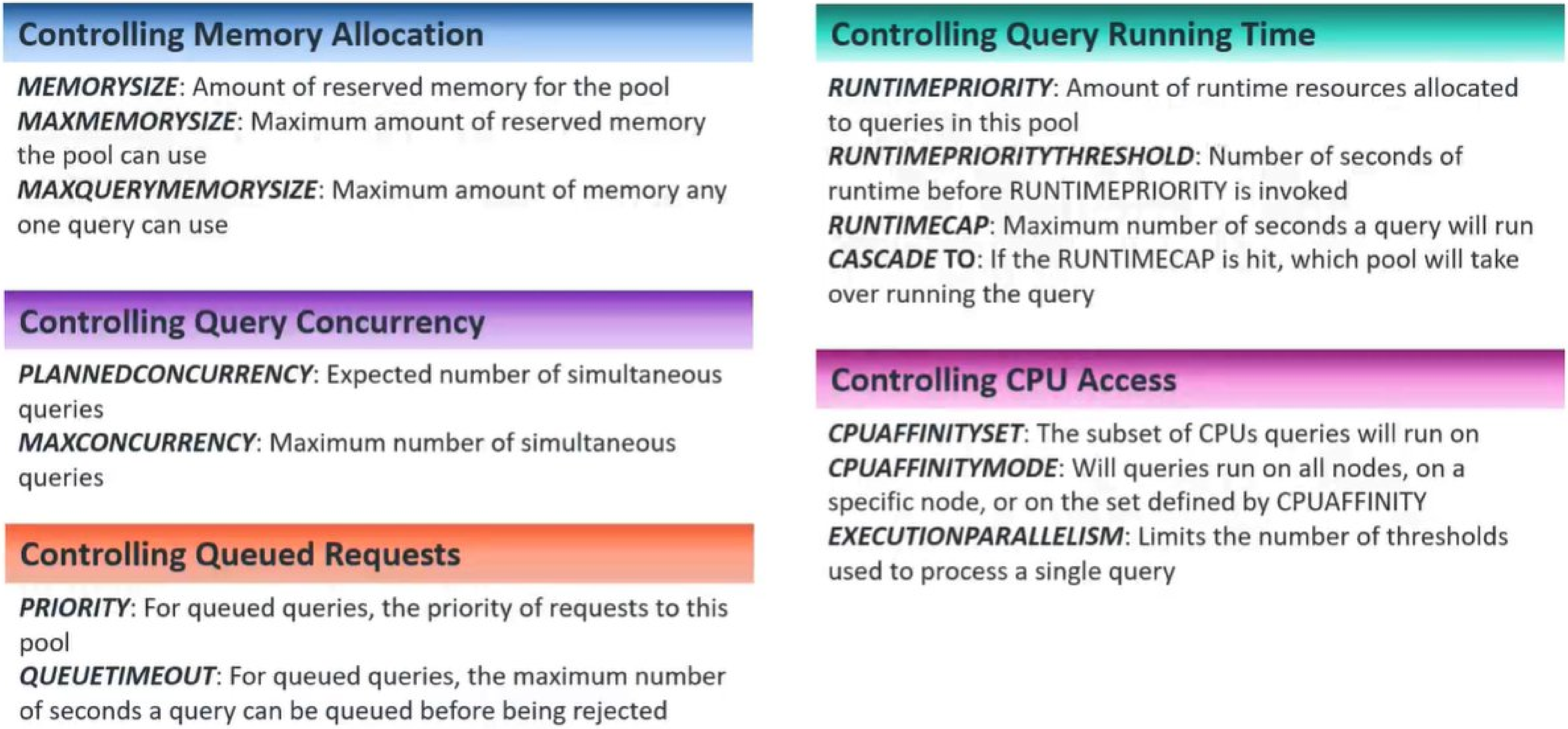
Query is **rejected**

Query Lifecycle and Resource Manager



opentext™

Resource Pool Parameters



Controlling Memory Allocation

MEMORYSIZE: Amount of reserved memory for the pool
MAXMEMORYSIZE: Maximum amount of reserved memory the pool can use
MAXQUERYMEMORYSIZE: Maximum amount of memory any one query can use

Controlling Query Concurrency

PLANNEDCONCURRENCY: Expected number of simultaneous queries
MAXCONCURRENCY: Maximum number of simultaneous queries

Controlling Queued Requests

PRIORITY: For queued queries, the priority of requests to this pool
QUEUETIMEOUT: For queued queries, the maximum number of seconds a query can be queued before being rejected

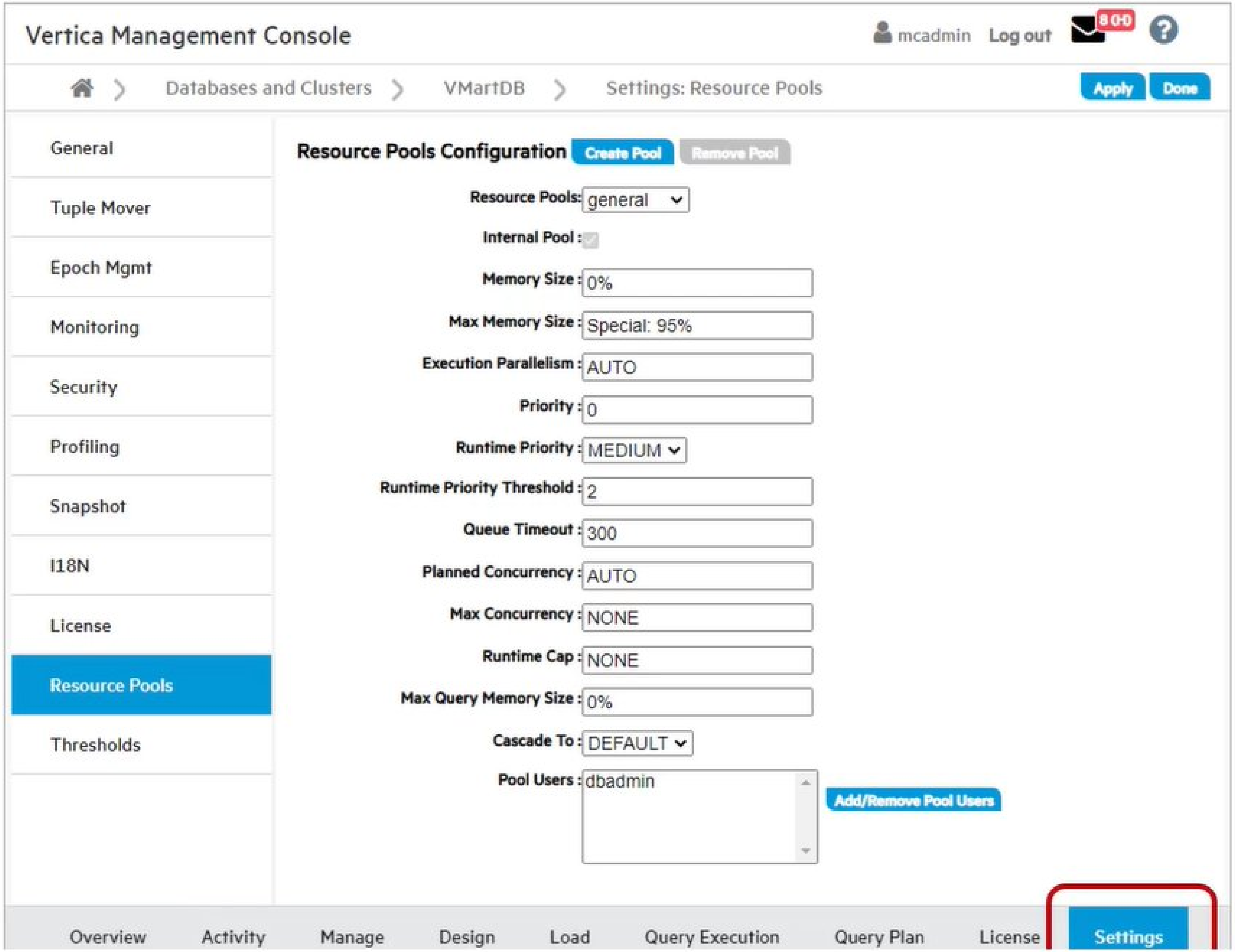
Controlling Query Running Time

RUNTIMEPRIORITY: Amount of runtime resources allocated to queries in this pool
RUNTIMEPRIORITYTHRESHOLD: Number of seconds of runtime before RUNTIMEPRIORITY is invoked
RUNTIMECAP: Maximum number of seconds a query will run
***CASCADE* TO**: If the RUNTIMECAP is hit, which pool will take over running the query

Controlling CPU Access

CPUAFFINITYSET: The subset of CPUs queries will run on
CPUAFFINITYMODE: Will queries run on all nodes, on a specific node, or on the set defined by CPUAFFINITY
EXECUTIONPARALLELISM: Limits the number of thresholds used to process a single query

Resource Pools in MC



Resource Pools Configuration

Resource Pools: general

Internal Pool : ☑

Memory Size : 0%

Max Memory Size : Special: 95%

Execution Parallelism : AUTO

Priority : 0

Runtime Priority : MEDIUM

Runtime Priority Threshold : 2

Queue Timeout : 300

Planned Concurrency : AUTO

Max Concurrency : NONE

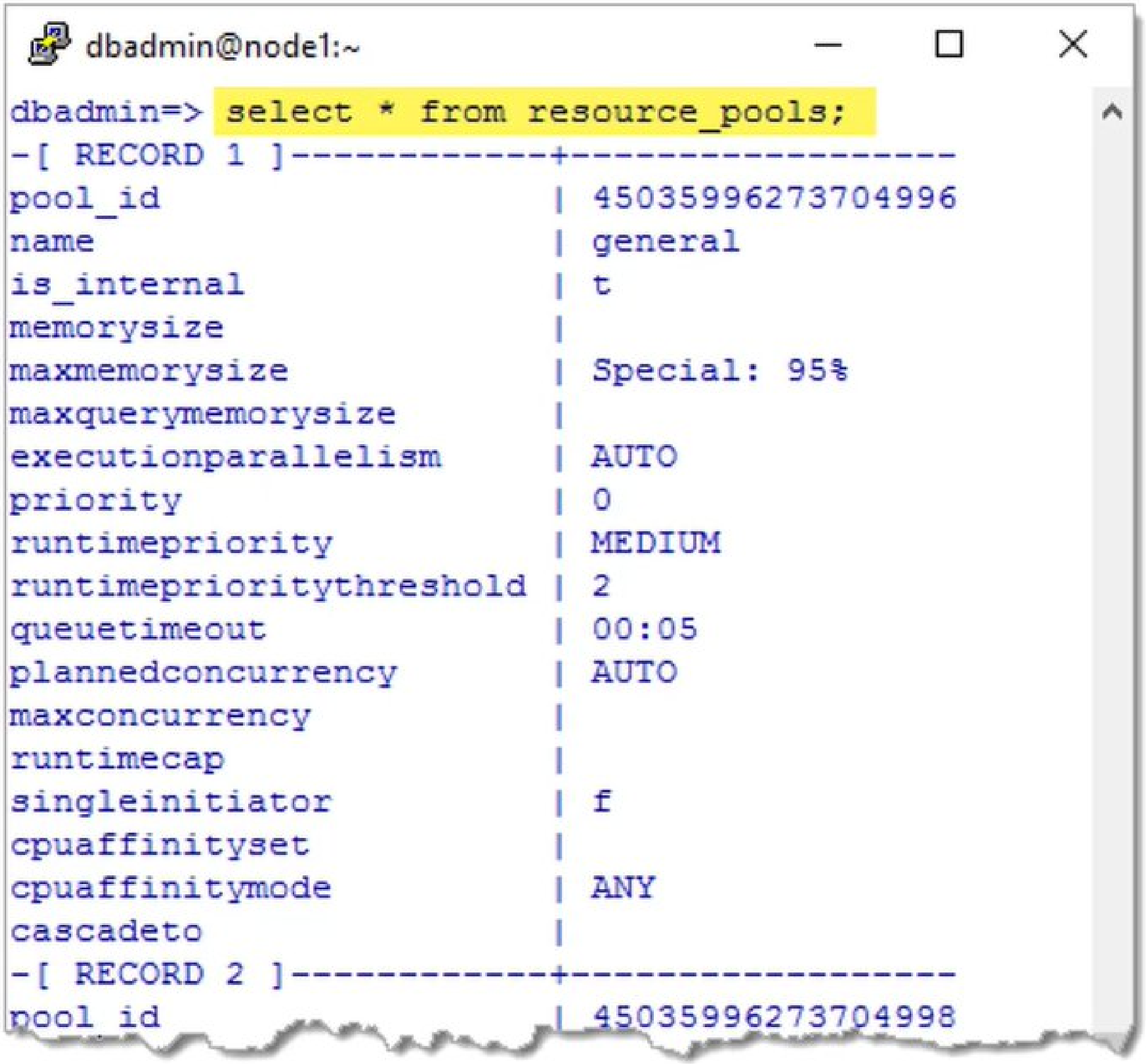
Runtime Cap : NONE

Max Query Memory Size : 0%

Cascade To : DEFAULT

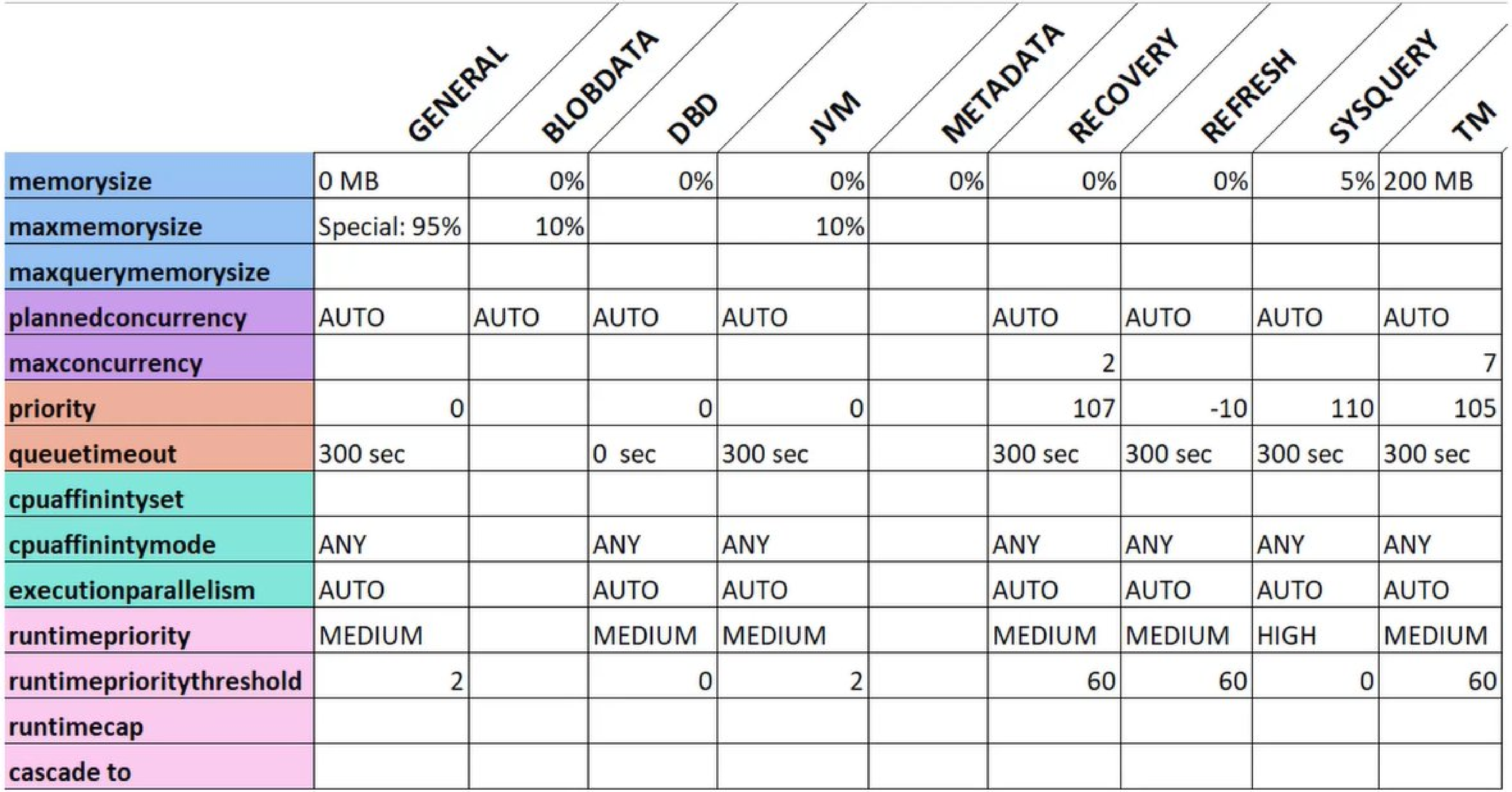
Pool Users : dbadmin

Resource Pools in vsql

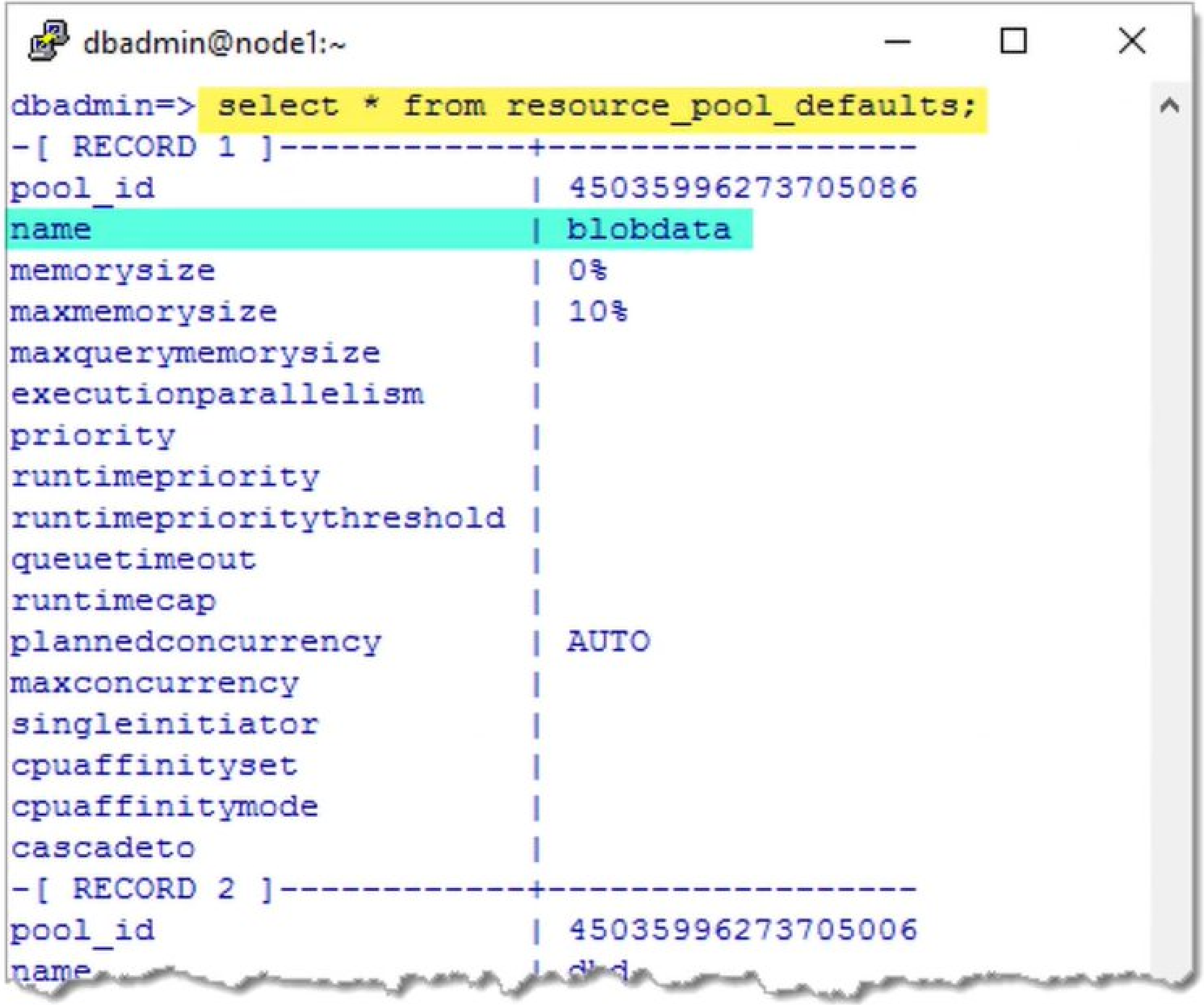


```
dbadmin=> select * from resource_pools;
-[ RECORD 1 ]------------+------------------
pool_id                  | 45035996273704996
name                     | general
is_internal              | t
memorysize               |
maxmemorysize            | Special: 95%
maxquerymemorysize       |
executionparallelism     | AUTO
priority                 | 0
runtimepriority          | MEDIUM
runtimeprioritythreshold | 2
queuetimeout             | 00:05
plannedconcurrency       | AUTO
maxconcurrency           |
runtimecap               |
singleinitiator          | f
cpuaffinityset           |
cpuaffinitymode          | ANY
cascadeto                |
-[ RECORD 2 ]------------+------------------
pool_id                  | 45035996273704998
```

Default Parameters



	GENERAL	BLOBDATA	DBD	JVM	METADATA	RECOVERY	REFRESH	SYSQUERY	TM
memorysize	0 MB	0%	0%	0%	0%	0%	0%	5%	200 MB
maxmemorysize	Special: 95%	10%		10%					
maxquerymemorysize									
plannedconcurrency	AUTO	AUTO	AUTO	AUTO		AUTO	AUTO	AUTO	AUTO
maxconcurrency						2			7
priority	0		0	0		107	-10	110	105
queuetimeout	300 sec		0 sec	300 sec		300 sec	300 sec	300 sec	300 sec
cpuaffinintyset									
cpuaffinintymode	ANY		ANY	ANY		ANY	ANY	ANY	ANY
executionparallelism	AUTO		AUTO	AUTO		AUTO	AUTO	AUTO	AUTO
runtimepriority	MEDIUM		MEDIUM	MEDIUM		MEDIUM	MEDIUM	HIGH	MEDIUM
runtimeprioritythreshold	2		0	2		60	60	0	60
runtimecap									
cascade to									

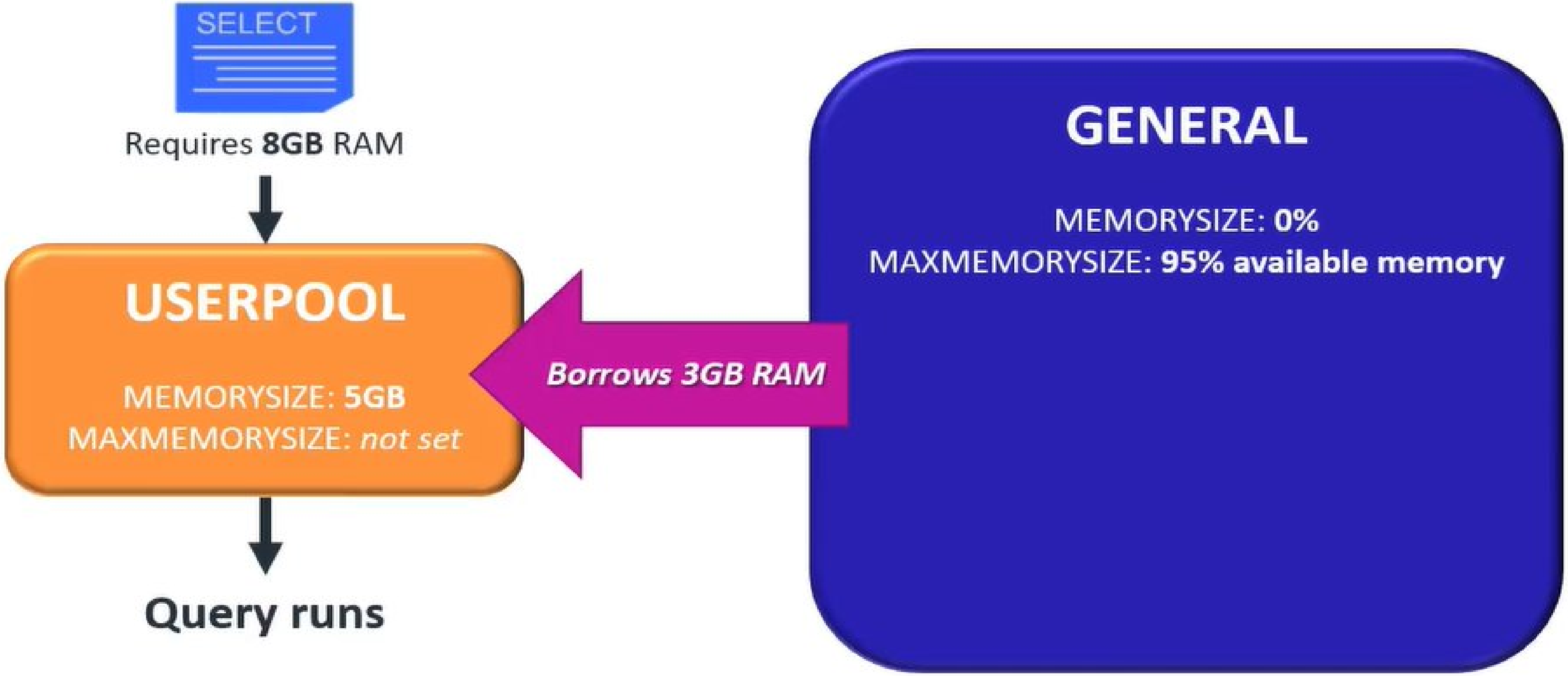


```
dbadmin=> select * from resource_pool_defaults;
-[ RECORD 1 ]------------+------------------
pool_id                  | 45035996273705086
name                     | blobdata
memorysize               | 0%
maxmemorysize            | 10%
maxquerymemorysize       |
executionparallelism     |
priority                 |
runtimepriority          |
runtimeprioritythreshold |
queuetimeout             |
runtimecap               |
plannedconcurrency       | AUTO
maxconcurrency           |
singleinitiator          |
cpuaffinityset           |
cpuaffinitymode          |
cascadeto                |
-[ RECORD 2 ]------------+------------------
pool_id                  | 45035996273705006
name                     | dbd
```

Memory Allocation

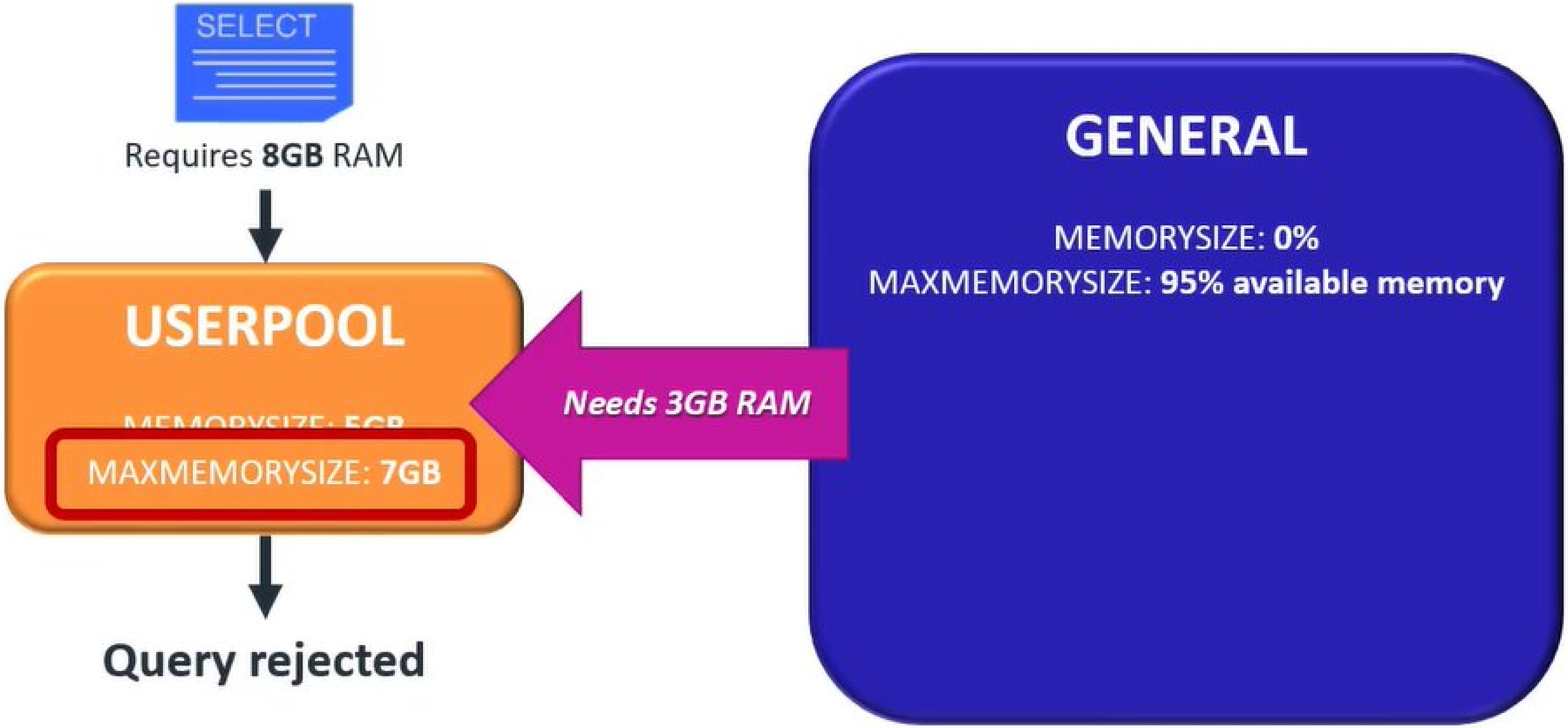
MEMORYSIZE: the amount of memory reserved for a pool

MAXMEMORYSIZE: the allowable amount of memory a pool can use



MEMORYSIZE: the amount of memory reserved for a pool

MAXMEMORYSIZE: the allowable amount of memory a pool can use



SELECT

Requires **8GB** RAM

USERPOOL

MEMORYSIZE: **5GB**

MAXMEMORYSIZE: **7GB**

Needs 3GB RAM

GENERAL

MEMORYSIZE: **0%**

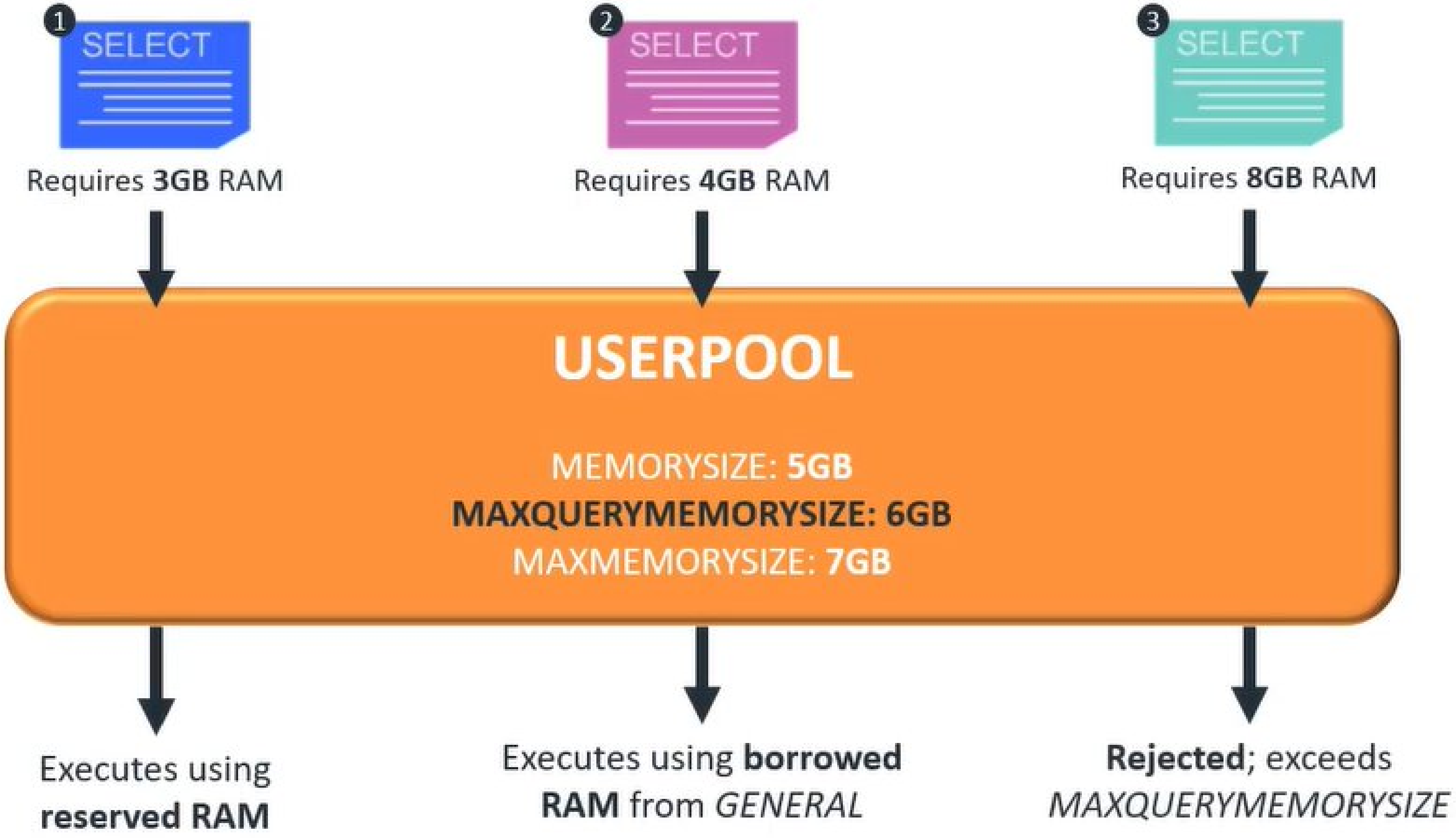
MAXMEMORYSIZE: **95% available memory**

Query rejected

MEMORYSIZE: the amount of memory reserved for a pool

MAXMEMORYSIZE: the allowable amount of memory a pool can use

MAXQUERYMEMORYSIZE: the maximum memory a query can use



Requires **3GB** RAM

Requires **4GB** RAM

Requires **8GB** RAM

USERPOOL

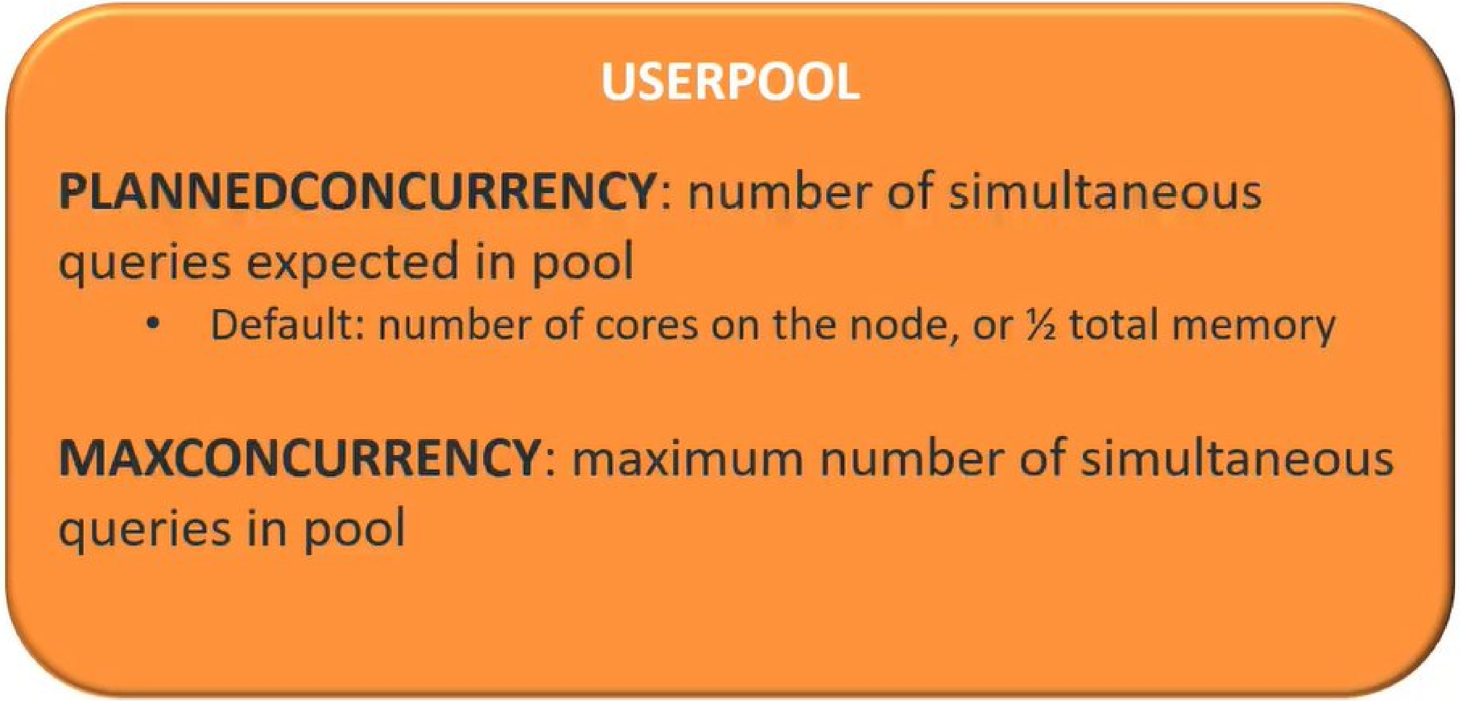
MEMORYSIZE: **5GB**
MAXQUERYMEMORYSIZE: 6GB
MAXMEMORYSIZE: **7GB**

Executes using **reserved RAM**

Executes using **borrowed RAM** from *GENERAL*

Rejected; exceeds *MAXQUERYMEMORYSIZE*

Query Concurrency



USERPOOL

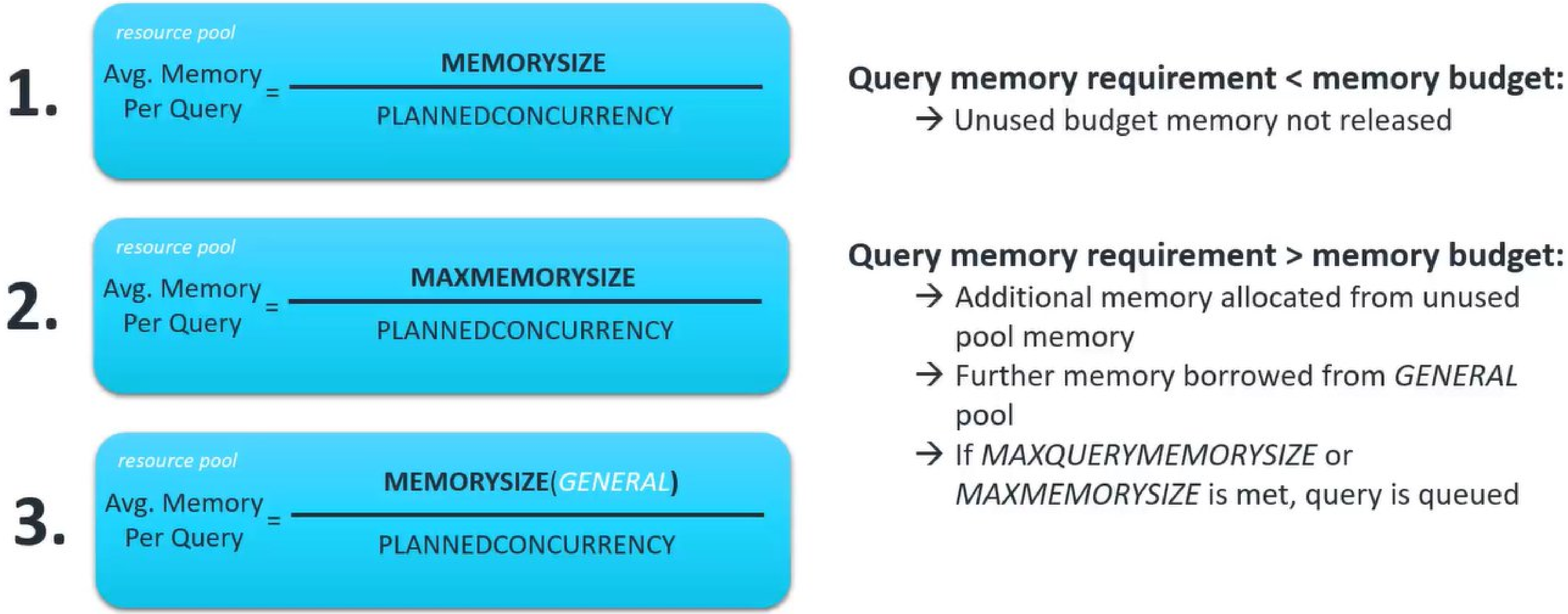
PLANNEDCONCURRENCY: number of simultaneous queries expected in pool

- Default: number of cores on the node, or ½ total memory

MAXCONCURRENCY: maximum number of simultaneous queries in pool

opentext™

Query Concurrency: Budgeting



1. *resource pool*

$$\text{Avg. Memory Per Query} = \frac{\textbf{MEMORYSIZE}}{\text{PLANNEDCONCURRENCY}}$$

2. *resource pool*

$$\text{Avg. Memory Per Query} = \frac{\textbf{MAXMEMORYSIZE}}{\text{PLANNEDCONCURRENCY}}$$

3. *resource pool*

$$\text{Avg. Memory Per Query} = \frac{\textbf{MEMORYSIZE}(\textit{GENERAL})}{\text{PLANNEDCONCURRENCY}}$$

Query memory requirement < memory budget:

→ Unused budget memory not released

Query memory requirement > memory budget:

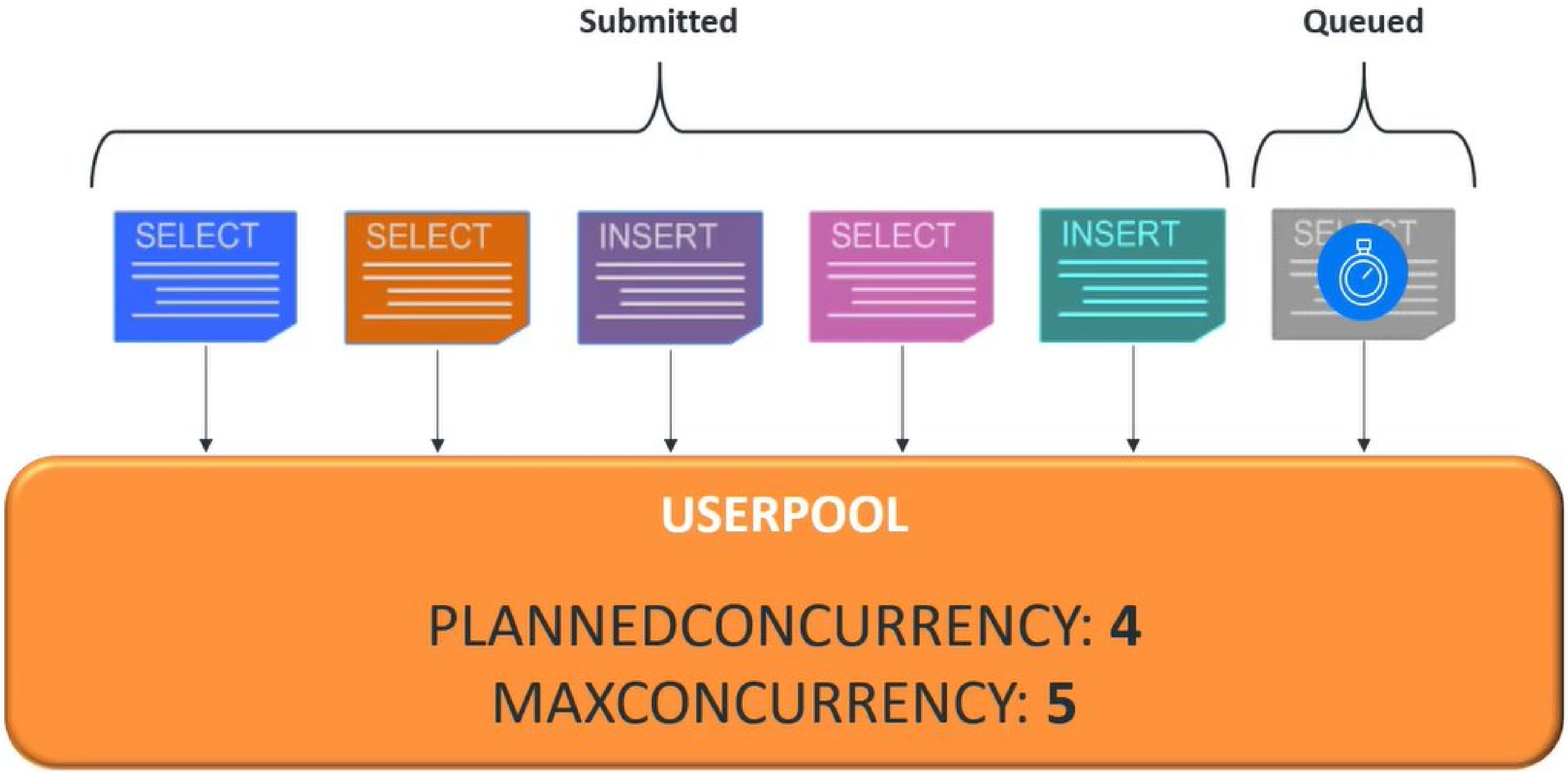
→ Additional memory allocated from unused pool memory

→ Further memory borrowed from *GENERAL* pool

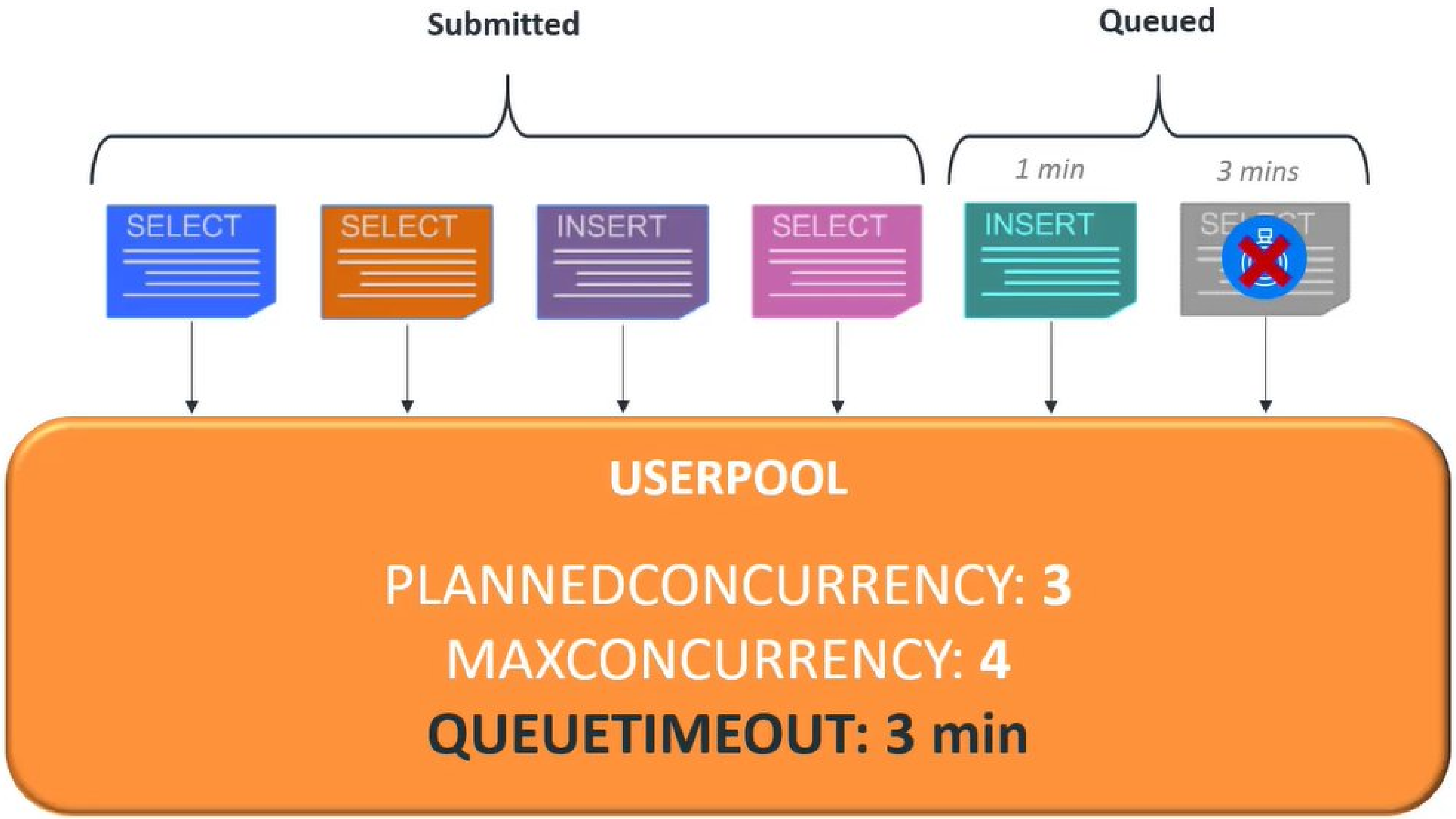
→ If *MAXQUERYMEMORYSIZE* or *MAXMEMORYSIZE* is met, query is queued

opentext™

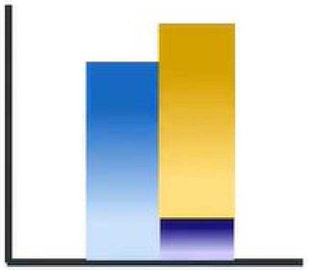
Query Concurrency: Simultaneous Queries



Query Concurrency: Queued Time Out



Query Requests across Pools

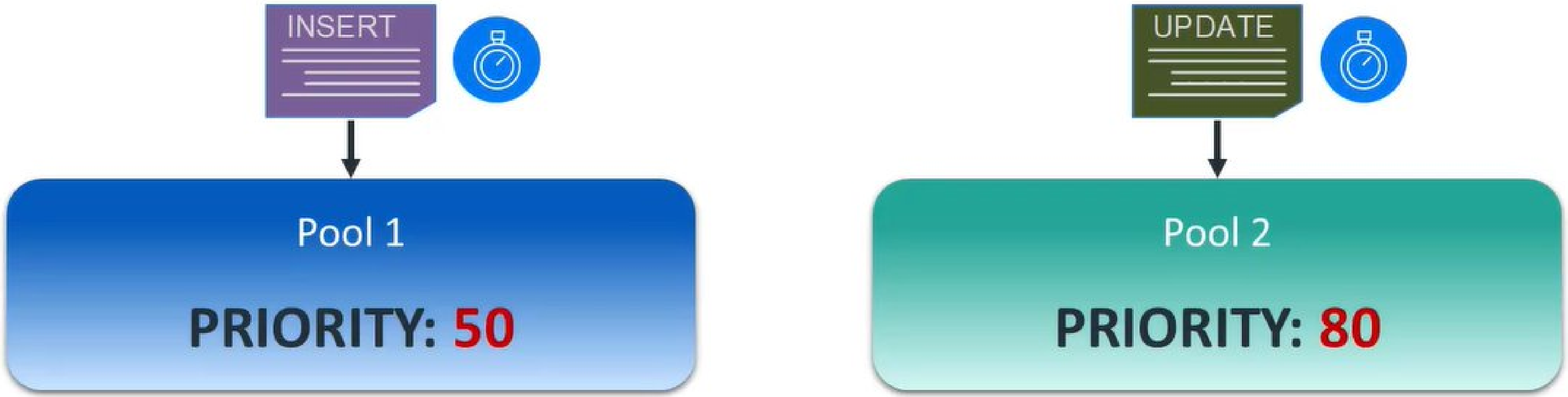


Query is **queued**

PRIORITY: When resources are constrained, determines which pool gets freed-up resources first

QUEUETIMEOUT: Maximum time a query can be queued before being cancelled

Scenario 1:



INSERT

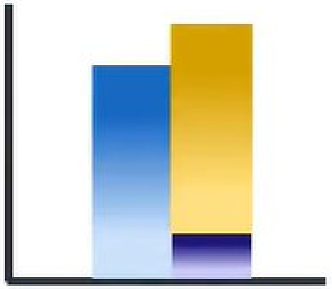
Pool 1

PRIORITY: 50

UPDATE

Pool 2

PRIORITY: 80

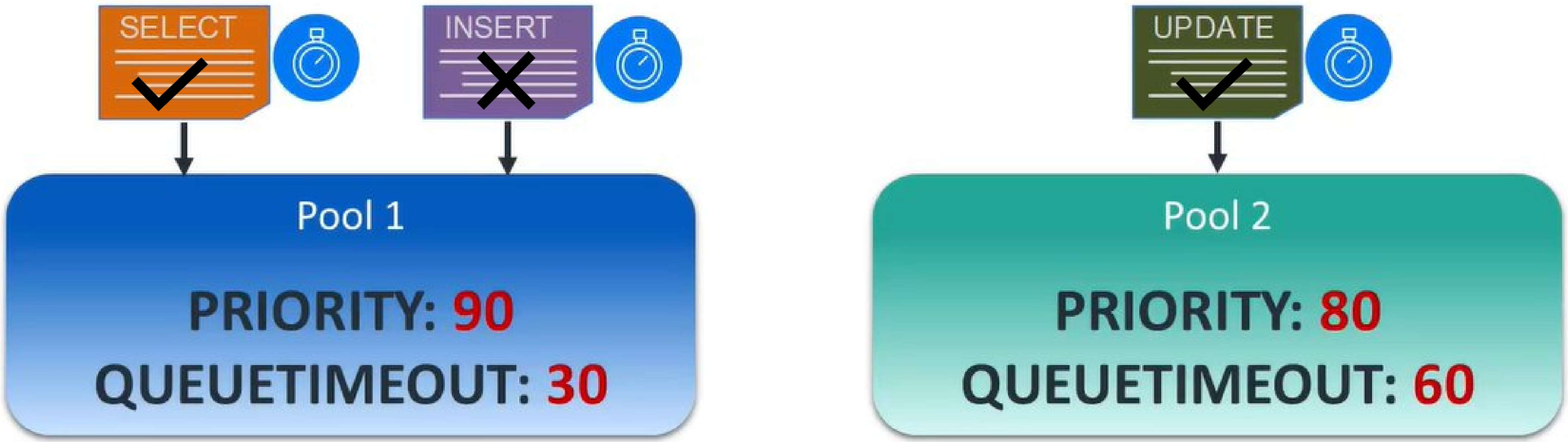


Query is **queued**

PRIORITY: When resources are constrained, determines which pool gets freed-up resources first

QUEUETIMEOUT: Maximum time a query can be queued before being cancelled

Scenario 2:



Pool 1

PRIORITY: 90

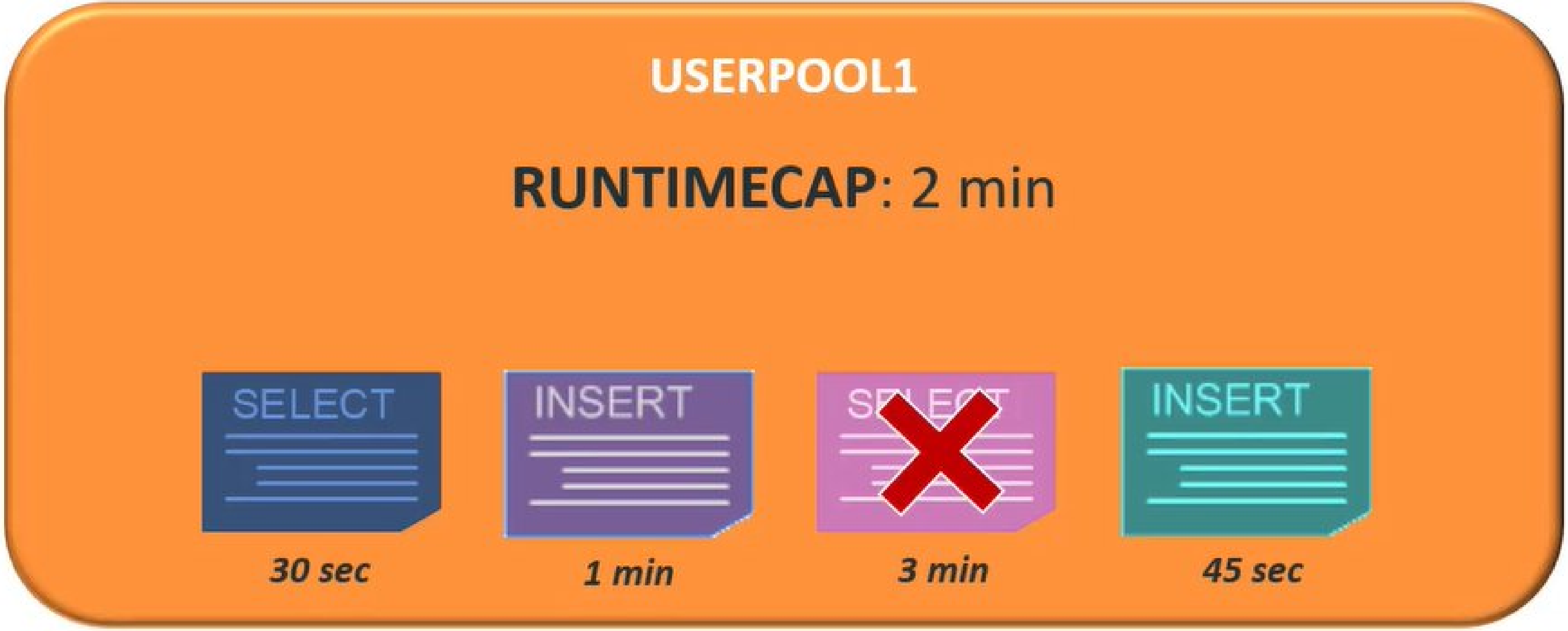
QUEUETIMEOUT: 30

Pool 2

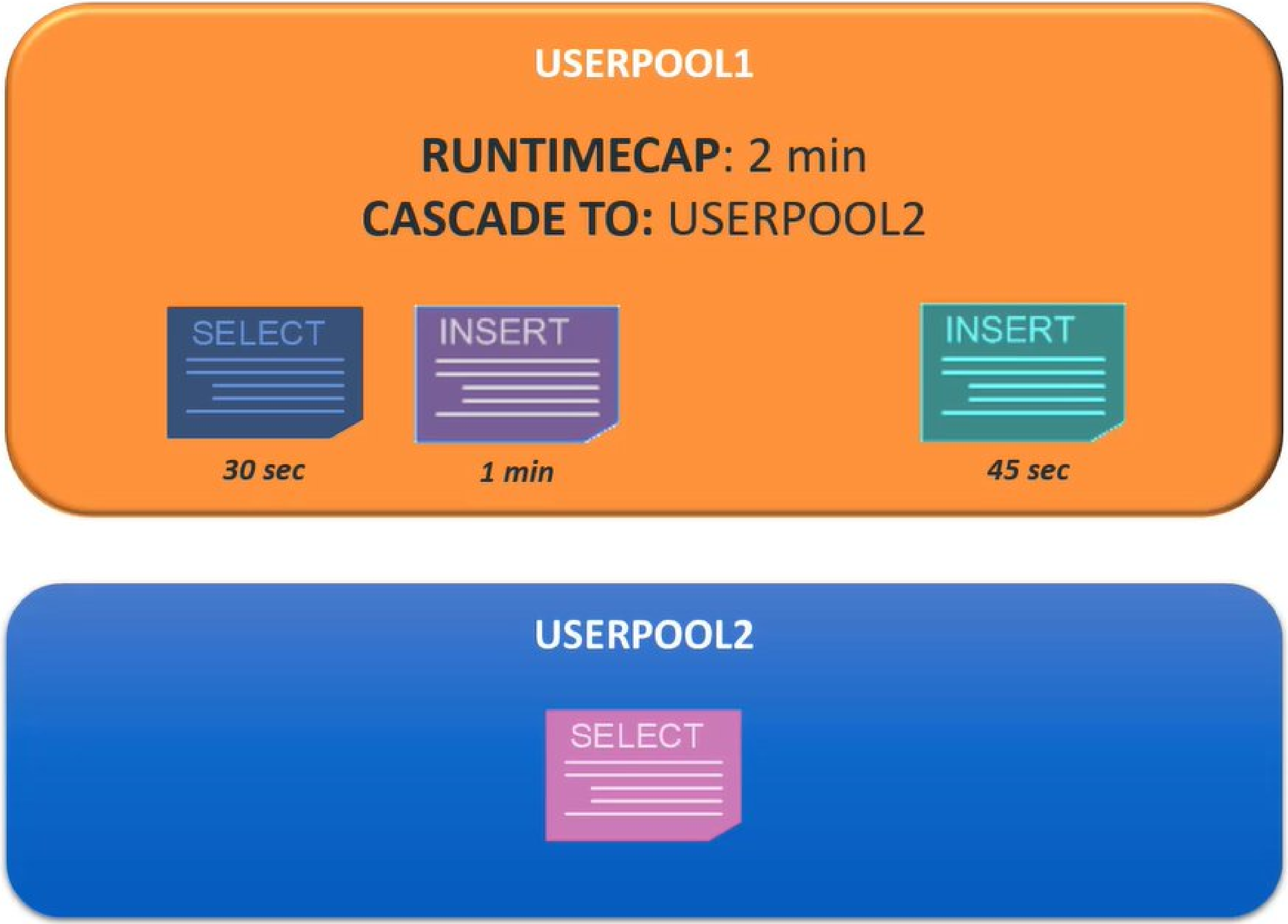
PRIORITY: 80

QUEUETIMEOUT: 60

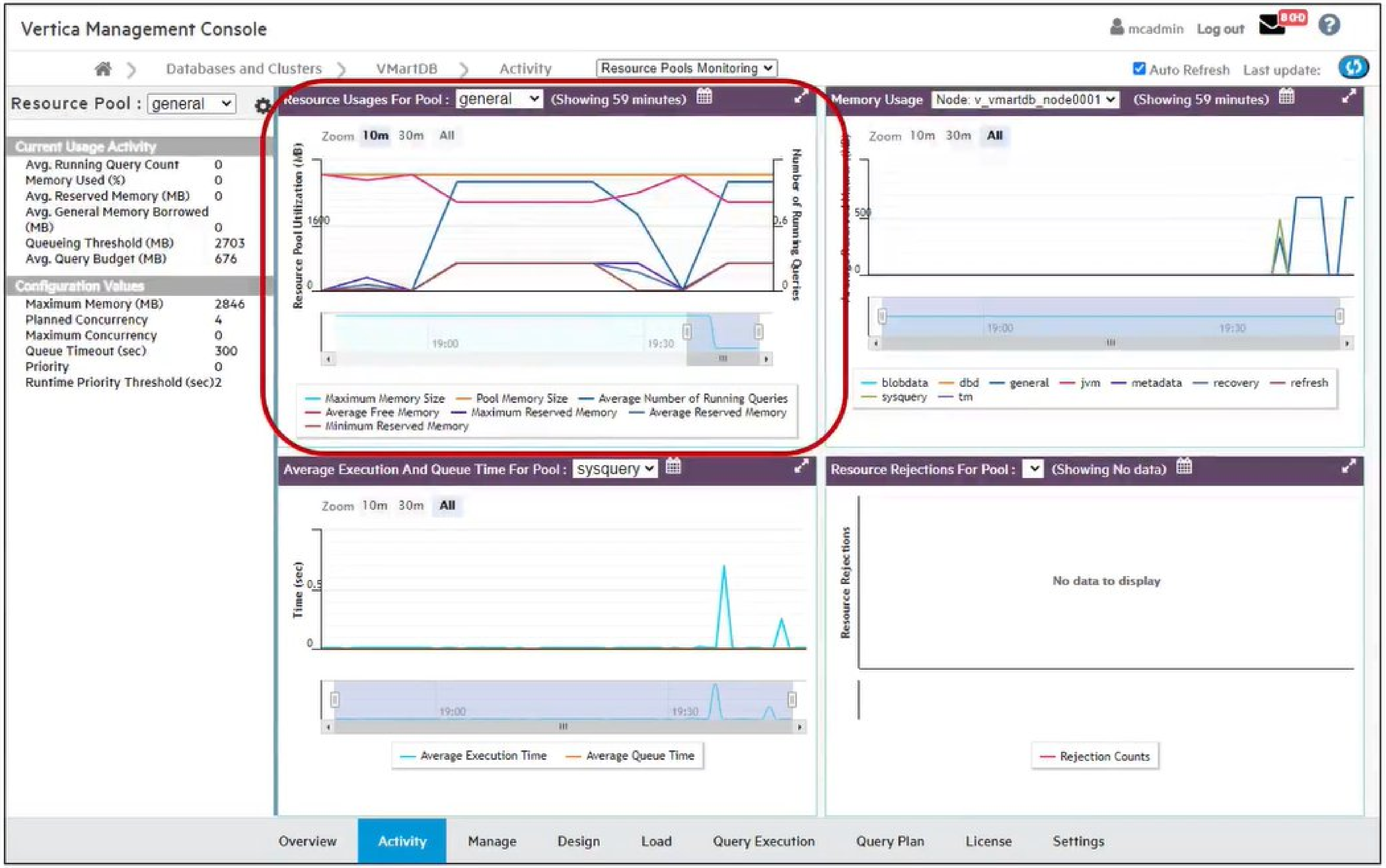
Query Execution Time



Cascade To: Avoid query rejections



Monitoring Resource Pools


Vertica Management Console

Databases and Clusters > VMartDB > Activity > Resource Pools Monitoring

Resource Pool : general

Current Usage Activity

Avg. Running Query Count	0
Memory Used (%)	0
Avg. Reserved Memory (MB)	0
Avg. General Memory Borrowed (MB)	0
Queueing Threshold (MB)	2703
Avg. Query Budget (MB)	676

Configuration Values

Maximum Memory (MB)	2846
Planned Concurrency	4
Maximum Concurrency	0
Queue Timeout (sec)	300
Priority	0
Runtime Priority Threshold (sec)	2

Resource Usages For Pool : general (Showing 59 minutes)

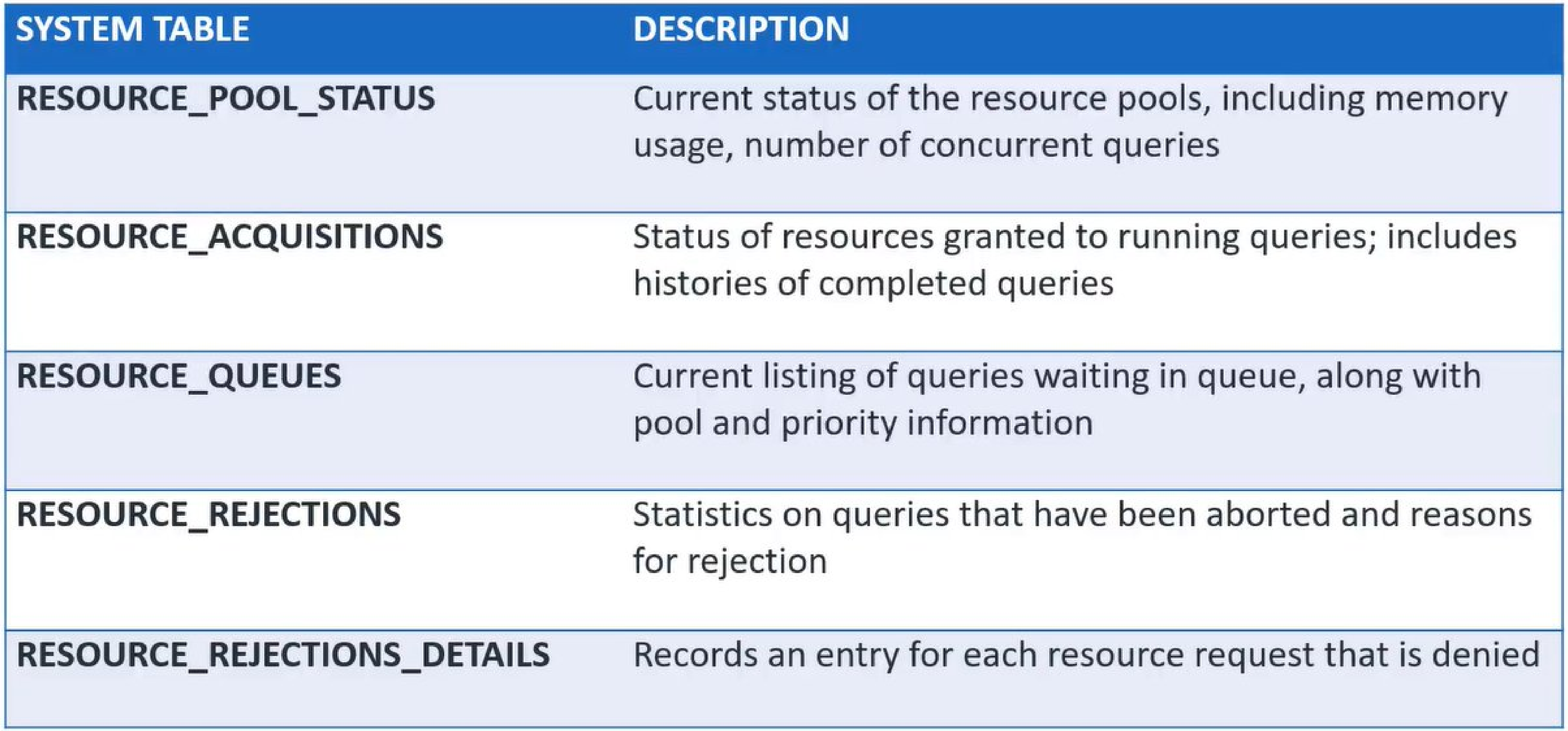
Memory Usage Node: v_vmartdb_node0001 (Showing 59 minutes)

Average Execution And Queue Time For Pool : sysquery

Resource Rejections For Pool : (Showing No data)

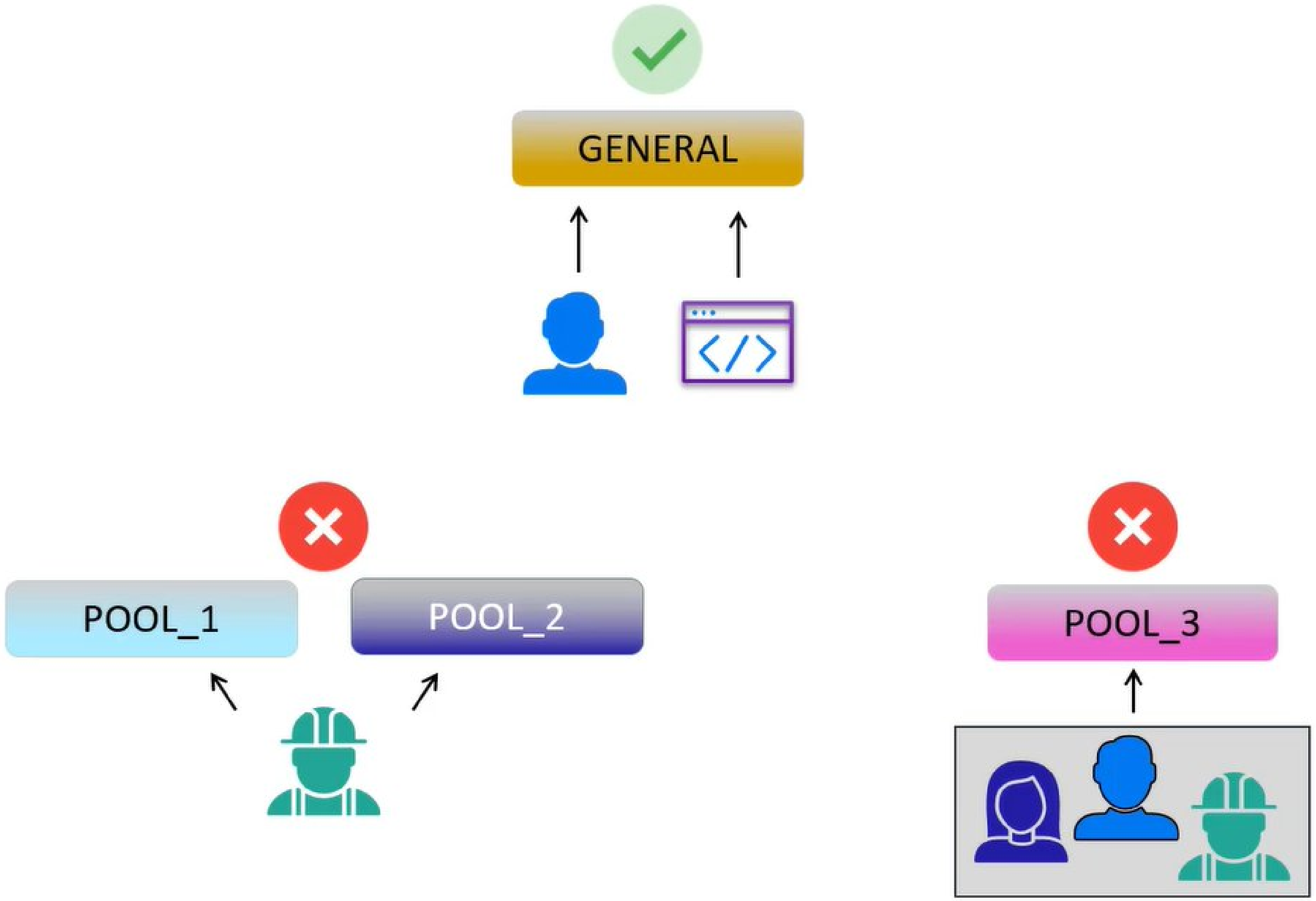
No data to display

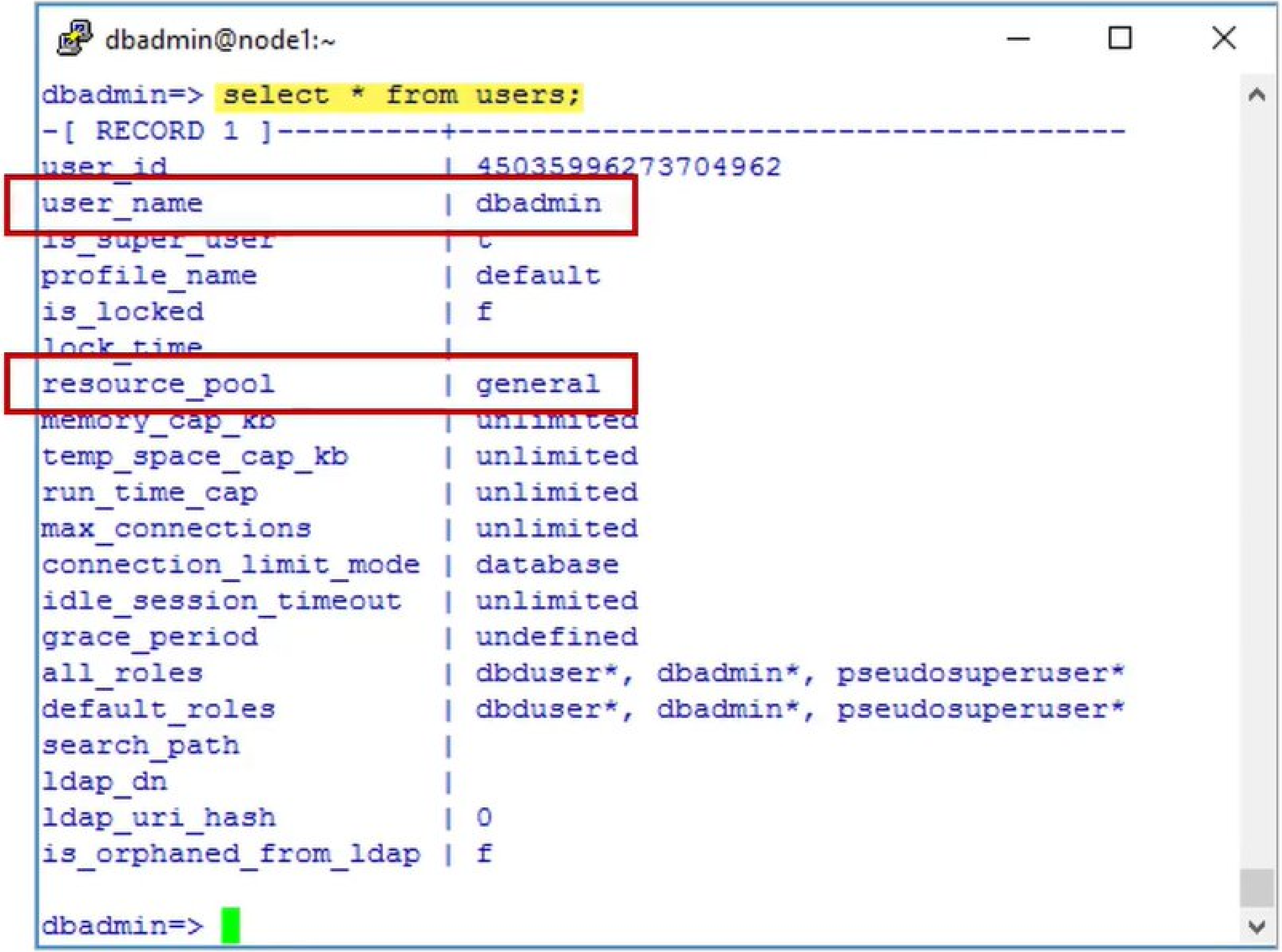
Monitoring Resource Pools



SYSTEM TABLE	DESCRIPTION
RESOURCE_POOL_STATUS	Current status of the resource pools, including memory usage, number of concurrent queries
RESOURCE_ACQUISITIONS	Status of resources granted to running queries; includes histories of completed queries
RESOURCE_QUEUES	Current listing of queries waiting in queue, along with pool and priority information
RESOURCE_REJECTIONS	Statistics on queries that have been aborted and reasons for rejection
RESOURCE_REJECTIONS_DETAILS	Records an entry for each resource request that is denied

Users and Resource Pools

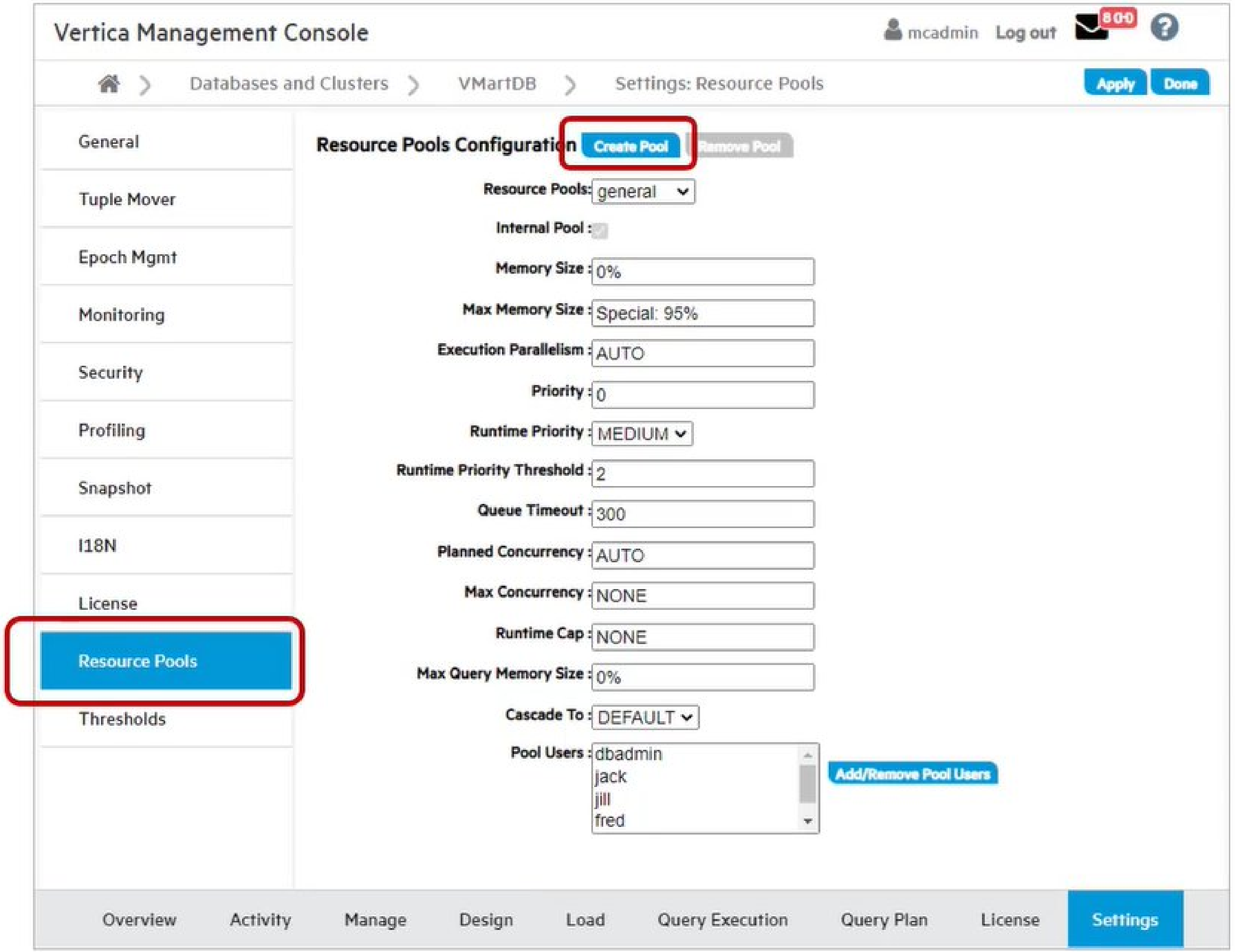




```
dbadmin=> select * from users;
-[ RECORD 1 ]---------+-------------------------------------
user_id               | 45035996273704962
user_name             | dbadmin
is_super_user         | t
profile_name          | default
is_locked             | f
lock_time             |
resource_pool         | general
memory_cap_kb         | unlimited
temp_space_cap_kb     | unlimited
run_time_cap          | unlimited
max_connections       | unlimited
connection_limit_mode | database
idle_session_timeout  | unlimited
grace_period          | undefined
all_roles             | dbduser*, dbadmin*, pseudosuperuser*
default_roles         | dbduser*, dbadmin*, pseudosuperuser*
search_path           |
ldap_dn               |
ldap_uri_hash         | 0
is_orphaned_from_ldap | f

dbadmin=>
```

Creating User Defined Resource Pools



Resource Pools Configuration

Resource Pools: general

Internal Pool : ☑

Memory Size : 0%

Max Memory Size : Special: 95%

Execution Parallelism : AUTO

Priority : 0

Runtime Priority : MEDIUM

Runtime Priority Threshold : 2

Queue Timeout : 300

Planned Concurrency : AUTO

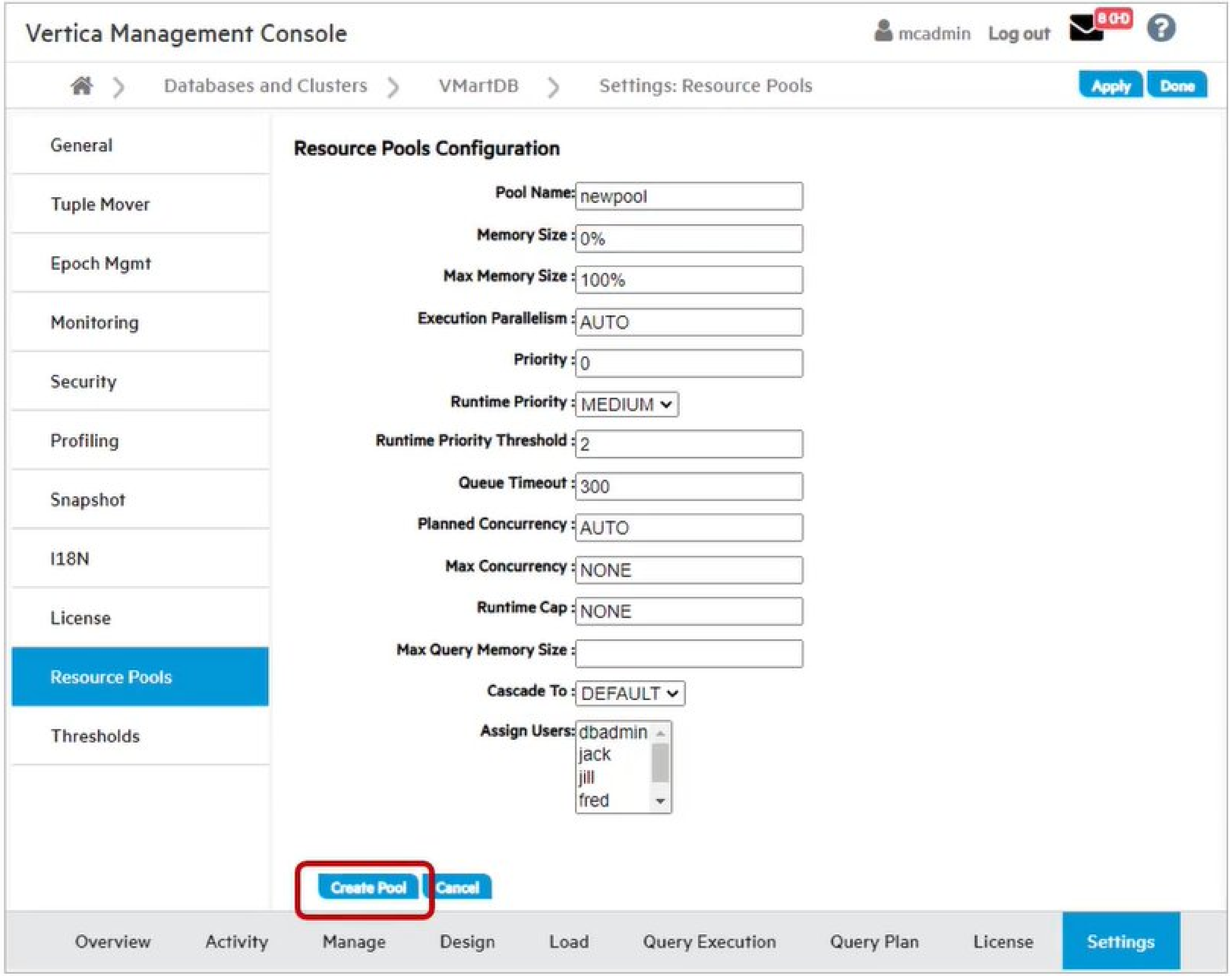
Max Concurrency : NONE

Runtime Cap : NONE

Max Query Memory Size : 0%

Cascade To : DEFAULT

Pool Users : dbadmin
jack
jill
fred



Resource Pools Configuration

Pool Name: newpool

Memory Size : 0%

Max Memory Size : 100%

Execution Parallelism : AUTO

Priority : 0

Runtime Priority : MEDIUM

Runtime Priority Threshold : 2

Queue Timeout : 300

Planned Concurrency : AUTO

Max Concurrency : NONE

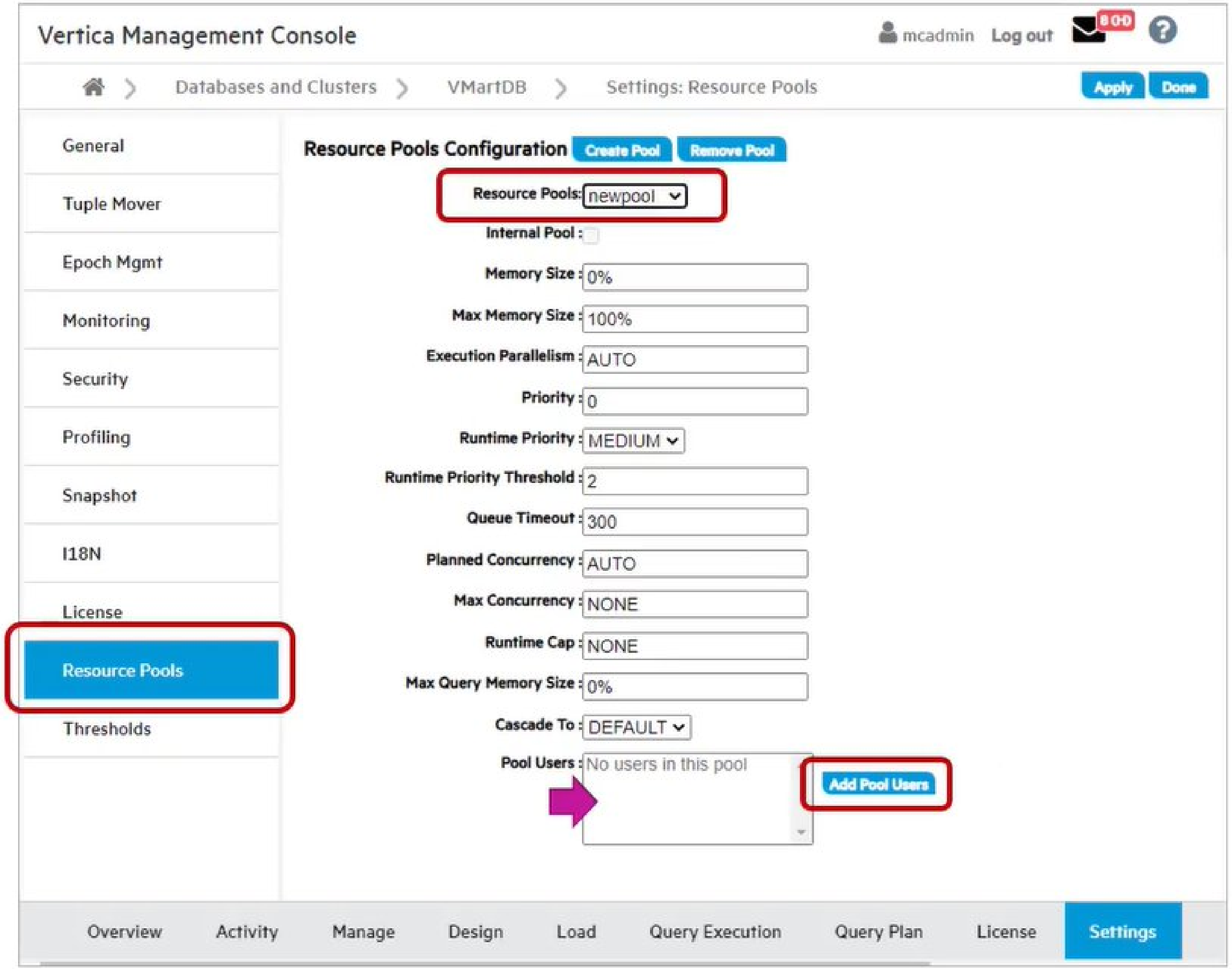
Runtime Cap : NONE

Max Query Memory Size :

Cascade To : DEFAULT

Assign Users: dbadmin
jack
jill
fred

Adding Users to Pools



Resource Pools Configuration

Resource Pools: newpool

Internal Pool : □

Memory Size : 0%

Max Memory Size : 100%

Execution Parallelism : AUTO

Priority : 0

Runtime Priority : MEDIUM

Runtime Priority Threshold : 2

Queue Timeout : 300

Planned Concurrency : AUTO

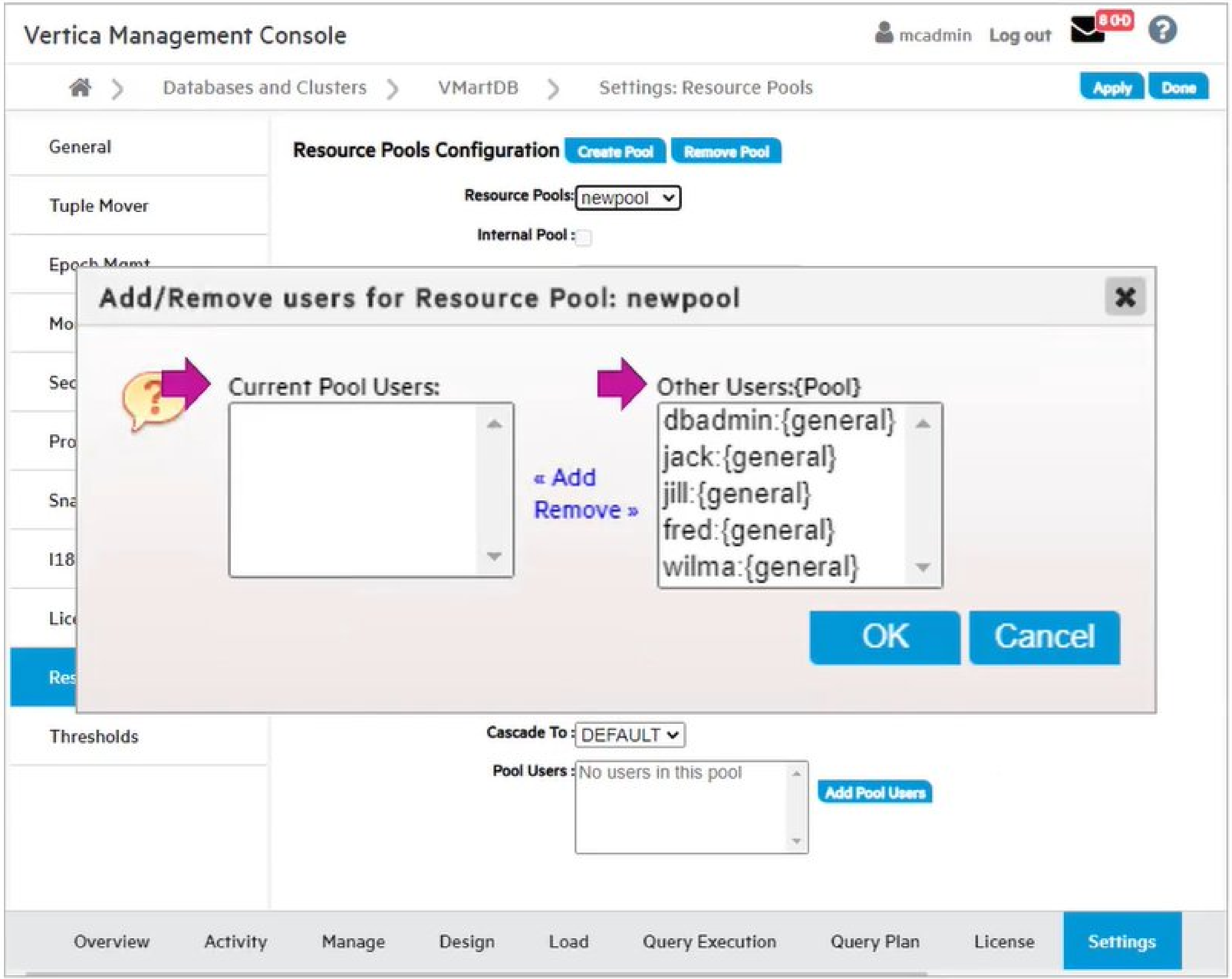
Max Concurrency : NONE

Runtime Cap : NONE

Max Query Memory Size : 0%

Cascade To : DEFAULT

Pool Users : No users in this pool



Resource Pools Configuration

Resource Pools: newpool

Internal Pool : □

Add/Remove users for Resource Pool: newpool

Current Pool Users:

« Add
Remove »

Other Users:{Pool}
dbadmin:{general}
jack:{general}
jill:{general}
fred:{general}
wilma:{general}

OK Cancel

Cascade To : DEFAULT

Pool Users : No users in this pool

Runtime Limit for Queries

You can set a limit for time a query is allowed to run. You can set this limit at three levels, listed in descending order of precedence:

The resource pool to which the user is assigned.

User profile with RUNTIMECAP configured by CREATE USER/ALTER USER

Session queries, set by SET SESSION RUNTIMECAP

```
=> CREATE USER user1 RESOURCE POOL ad_hoc_queries;

=> ALTER USER user1 RUNTIMECAP '60 minutes';

=> ALTER RESOURCE POOL ad_hoc_queries RUNTIMECAP '30 minutes';
```

Periodic Batch Loads

Scenario

You do batch loads every night, or occasionally (infrequently) during the day. When loads are running, it is acceptable to reduce resource usage by queries, but at all other times you want all resources to be available to queries.

Solution

Create a separate resource pool for loads with a higher priority than the preconfigured setting on the build-in GENERAL pool.

In this scenario, nightly loads get preference when borrowing memory from the GENERAL pool. When loads are not running, all memory is automatically available for queries.

```
=> CREATE RESOURCE POOL load_pool PRIORITY 10;
=> ALTER USER load_user RESOURCE POOL load_pool;
```

Critical Query

Scenario

The CEO runs a report every Monday at 9AM, and you want to be sure that the report always runs.

Solution

To ensure that a certain query or class of queries always gets resources, you could create a dedicated pool for it as follows:

1. Using the PROFILE command, run the query that the CEO runs every week to determine how much memory should be allocated

2. At the end of the query, the system returns a notice with resource usage with a hint:

*NOTICE: Statement is being profiled.HINT: select * from v_monitor.execution_engine_profiles where transaction_id=45035996273751349 and statement_id=6; NOTICE: Initiator memory estimate for query: [on pool general: 1723648 KB, minimum: 355920 KB]*

3. Create a resource pool with MEMORYSIZE reported by the above hint to ensure that the CEO query has at least this memory reserved for it.

=> ALTER USER ceo_user RESOURCE POOL ceo_pool;

Preventing Runaway Queries

Scenario

Joe, a business analyst often runs big reports in the middle of the day that take up the whole machine's resources. You want to prevent Joe from using more than 100MB of memory, and you want to also limit Joe's queries to run for less than 2 hours.

Solution

User Profiles provides a solution to this scenario. To restrict the amount of memory Joe can use at one time, set a MEMORYCAP for Joe to 100MB using the ALTER USER command. To limit the amount of time that Joe's query can run, set a RUNTIMECAP to 2 hours using the same command. If any query run by Joe takes up more than its cap, Vertica rejects the query.

If you have a whole class of users whose queries you need to limit, you can also create a resource pool for them and set RUNTIMECAP for the resource pool. When you move these users to the resource pool, Vertica limits all queries for these users to the RUNTIMECAP you specified for the resource pool

```
=> ALTER USER analyst_user MEMORYCAP '100M' RUNTIMECAP '2 hours';
```

Continuous Load and Query

Scenario

You want your application to run continuous load streams, but many have up concurrent query streams. You want to ensure that performance is predictable.

Solution

The solution to this scenario depends on your query mix. In all cases, the following approach applies:

1.Determine the number of continuous load streams required. Create a dedicated resource pool for the loads and associate it with the database user that will perform them.

2.In general, concurrency settings for the load pool should be less than the number of cores per node. Unless the source processes are slow, it is more efficient to dedicate more memory per load and have additional loads queue. Adjust the load pool's QUEUETIMEOUT setting if queuing is expected.

2.Run the load workload for a while and observe whether the load performance is as expected. If the Tuple Mover is not tuned adequately to cover the load behavior, see Managing the Tuple Mover in Administrator's Guide.

3.If there is more than one kind of query in the system—some queries must be answered quickly, while others are part of a batch reporting process—follow the guidelines in Handling Mixed Workloads: Batch versus Interactive.

4.Let the queries run and observe performance. If some classes of queries do not perform as desired, then you might need to tune the GENERAL pool as outlined in Restricting Resource Usage of Ad Hoc Query Application or create more dedicated resource pools for those queries.

Thank you