Technical Report

# OnCommand Unified Manager Schemas Explained

Dhiman Chakraborty, NetApp
August 2018 | TR-4709

Version 1.0

## Abstract

This technical report discusses all the Unified Manager schemas exposed from NetApp® OnCommand® Unified Manager 7.3 and later. It describes the use of various read-only schemas with examples to help end users create content-rich custom reports. Readers should have some knowledge of SQL constructs and should know how to create and integrate custom reports.

NetApp®

## TABLE OF CONTENTS



## LIST OF TABLES



## LIST OF FIGURES



# 1 Purpose

This document is intended to help users create content-rich custom capacity and performance reports with the reporting tool of their choice. With the open database approach, users have access to almost all the schemas of the Unified Manager database. This document guides you in understanding all the schemas and how each one helps you generate granular data in your NetApp ONTAP® systems, whether in health, capacity, or performance. The document includes a detailed section on performance schemas with examples to help you meet your operational performance reporting goals.

# 2 Introduction

OnCommand Unified Manager uses a MySQL database to record data. Both the Unified Manager and the ocie collector persist data into the various schemas in the MySQL database. Users can interact with the schemas over a JDBC connection to create custom reports via several Integrated Development Environment (IDE) that allow a JDBC connection to a MySQL database. These schemas provide read-only access to the Unified Manager database so that users can query the schema for relevant Unified Manager objects they intend to create custom reports about.

Until Unified Manager 7.2, users had access to only one such schema, `ocum_report`. This created limitations in the granularity of custom reports that users could create with any third-party tool via a JDBC connection. With Unified Manager 7.3, NetApp has exposed almost all of the schemas, making it easy to create a richer and more granular report.

The Report Schema role is the primary requirement for a database user to access the Unified Manager schemas. A database user with the Report Schema role must be created from the Unified Manager GUI. This user has only read-only permission for most tables in MySQL. Access is barred to any tables that hold user data or cluster credential information. All tables of the following schemas are exposed to the user with only the SELECT option.

```
mysql> show databases;
+------------------------+
| Database               |
+------------------------+
| netapp_model_view      |
| netapp_performance     |
| ocum                   |
| ocum_report            |
| ocum_report_birt       |
| opm                    |
| scalemonitor           |
+------------------------+
9 rows in set (0.00 sec)
```

## 2.1 Prerequisites

Users should have a beginner to intermediate level of knowledge in writing SQL queries and how OnCommand Unified Manager works.

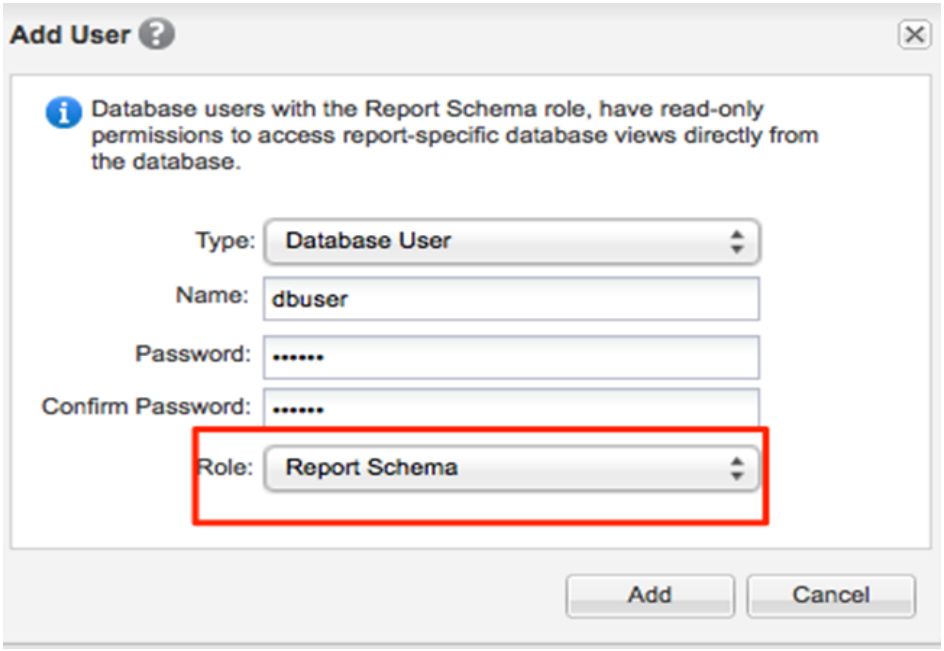
To understand the Unified Manager database custom reporting structure, read the technical report OnCommand Unified Manager Reporting

## 2.2 Accessing the Unified Manager MySQL Database

You can use your favorite software to create the SQL queries; for example, Eclipse, Tableau, Mysqladmin, and so on. However, for execution, the software needs to access the read-only schemas via a Unified Manager database user with the Report Schema role.

To create a database user in the Unified Manager server, follow these steps.

1. Log in to Unified Manager as the OnCommand administrator.
2. Navigate to Administration → Manage Users.
3. Click Add and add a database user with Report Schema as the role definition.




You are now ready to connect to the database.

**Note:** NetApp recommends not scheduling complex SQL queries to run at very frequent intervals, such as one query every 15 minutes. Scheduling too frequently can lead to performance degradation of your Unified Manager instance.

# 3 Unified Manager Schemas Explained

Each Unified Manager schema contains specific information. Some just deal with monitored object inventory, health, and capacity, while others deal with performance and quality of service (QoS). When creating a complex report that may need to draw health, performance, protection, and/or capacity information into a single report, you need to know the role of each schema and its contents. Although all schemas are available for external reference, you will touch upon some schemas more often than others. The following sections describe schemas that are most important in creating such a report.

## 3.1 netapp_model_view

The `netapp_model_view` database is where ocie puts data from ONTAP controllers. These tables are accessed directly only by the ocie code. The `netapp_model` database contains the database views that ocie publishes for the Unified Manager applications to read the data populated by the ocie collection.

These are read-only views that are accessed by the UM code. Tables from this table can be aggregate, cluster, disk, LUN, qtree, volume, storage virtual machine, and so on.

As mentioned earlier, report users can read the exposed schemes. Here are two sample queries the user can run.

**Example:** To find all RAID4 aggregates in a cluster, use the following query.

```
mysql>select name from aggregate where raidType='RAID4' and clusterId=1;
+--------------------------+
| name                     |
+--------------------------+
| aggr0_A700_141_11_12_01  |
| aggr0_A700_141_11_12_02_0 |
+--------------------------+
2 rows in set (0.00 sec)
```

**Example:** To find all volumes in a particular cluster, use the following query.

```
mysql> select name from volume where clusterId=1;
+-----------------------------------------+
| name                                    |
+-----------------------------------------+
| vol0                                    |
| vol0                                    |
| svm_root                                |
| A_dt_v1                                 |
| MDV_CRS_6ca54b7354bb11e799b400a09890ee89_A |
| MDV_CRS_6ca54b7354bb11e799b400a09890ee89_B |
+-----------------------------------------+
6 rows in set (0.00 sec)
```

## 3.2 ocum

The `ocum` database is where Unified Manager stores application data. It includes views that are used to support UI filtering and sorting and the calculation of some derived fields. A variety of types of data can be stored in these tables. Some of these tables contain additional fields for objects collected by ocie.

These *sidecar tables* are joined to the corresponding ocie table to define the Unified Manager object entity. Examples include aggregate, cluster, and volume tables. These tables hold Unified Manager specific values, such as the healthStatus, riskLevels, and isSnapMirrorDestination fields. The sidecar tables sometimes hold data that is interpreted from the data that ocie collects.

For example, in the volume table, the dfSnapshotBytesUsed and dfSnapshotBytesAvail columns are recalculated for certain volumes. The sidecar tables are populated and updated as part of the Unified Manager reconciliation process. Some of the tables contain persistent historical data. These history tables have names like aggregatehistorymonth, aggregatehistoryweek, and aggregatehistoryyear. They are updated by the Unified Manager history consolidators.

Some of the tables (aggregategrowthrateinfo and volumegrowthrateinfo) hold growth rate information. The columns in these table are values used internally to speed up growth rate calculations. Other tables

contain Unified Manager specific objects such as alert, event, and storage service. Certain tables are used for persisting support data for the Unified Manager platform components such as the job engine (job, task*, and readytaskworkitemqueueentry tables), Quartz scheduling (qrtz_* tables), and user management (role, authorizationunit). Most of the grid list views in the UI are populated using a database view whose name ends in livelistdtoview. These views join in all the tables necessary to populate all the columns in the live list grid and are also used to filter and sort the grid.

**Example:** The following query fetches dfSnapshotBytesUsed and dfSnapshotBytesAvail for all the volumes of a cluster.

**Note:** All metric values are in bytes.

```
mysql> select id, cluster_id, dfSnapshotBytesUsed, dfSnapshotBytesAvail from volume
where cluster_id = 1;

+-----+------------+--------------------+----------------------+
| id  | cluster_id | dfSnapshotBytesUsed | dfSnapshotBytesAvail |
+-----+------------+--------------------+----------------------+
| 419 |          1 |            14364672 |             39321600 |
| 421 |          1 |         38079754240 |                    0 |
+-----+------------+--------------------+----------------------+
2 rows in set (0.00 sec)
```

## 3.3 ocum_report and ocum_report_birt

These two schemas store data necessary for inventory configuration and capacity-related information. To obtain historical data for Unified Manager objects related to capacity, query this schema.

**Example:** The following query helps to understand volume relationships.

```
mysql> select id,
relationshipType,relationshipState,overallStatus,sourceVolumeName,destinationVolumeNam
e from volumerelationships;

+------+--------------------------+-------------------+--------------+---------------
---+---------------------------------------+
| id   | relationshipType         | relationshipState | overallStatus |
sourceVolumeName | destinationVolumeName             |
+------+--------------------------+-------------------+--------------+---------------
---+---------------------------------------+
| 3285 | extended_data_protection | snapmirrored      | NORMAL        | vs_srv_vol1
| vs_src_vs_srv_vol1_SnapMirror_001 |
| 3321 | extended_data_protection | broken-off          | ERROR           |
vs_srv_vol2      | vs_src_vs_srv_vol2_SnapMirror_001 |
+------+--------------------------+-------------------+--------------+---------------
---+-----------------------------------+
2 rows in set (0.00 sec)
```

## 3.4 netapp_performance

This is one of the most important databases in which opm and ocie store performance counters and QoS metrics. Most of the tables are cluster-specific. Tables from this schema include summary_vserver, summary_node, sample_cluster, summary_cluster, and so on.

```
mysql> select * from sample_cluster where clusterMaxAggregateUtilization > 25;
+-------+---------------+-------+------------+------------------+--------------------
-------+---------------------------------+
| objid | time          | empty | clusterOps | clusterThroughput |
clusterMaxNodeUtilization | clusterMaxAggregateUtilization |
+-------+---------------+-------+------------+------------------+--------------------
-------+---------------------------------+
|  1328 | 1498863872317 |     0 |          0 |            702633 |
15.2001 |                          25.7194 |
|  1328 | 1499294345647 |     0 |          0 |            698056 |
22.264 |                         25.9311 |
|  1328 | 1499294645647 |     0 |          0 |            681074 |
22.9006 |                            25.79 |
|  1328 | 1499294945647 |     0 |          0 |            712344 |
23.7666 |                          25.0478 |
|  1328 | 1499295249330 |     0 |          0 |            412075 |
17.3897 |                          25.6334 |
+-------+---------------+-------+------------+------------------+--------------------
-------+---------------------------------+
5 rows in set (0.00 sec)
```

Because of the type of data, this database is one of the most complex and difficult to understand. The performance schema is discussed in detail in Section 4.

## 3.5 opm

This schema has opm-specific configuration settings. It also contains all threshold information.

Tables from this schema include acquisition_info, polling_interval_history, ext_server_config, and so on.

**Example:** Use the following query to check polling interval history.

```
mysql> select * from polling_interval_history;
+------------+--------------+
| changeTime | pollInterval |
+------------+--------------+
|          0 |            5 |
+------------+--------------+
1 row in set (0.00 sec)
```

## 3.6 Scalemonitor

Scalemonitor gives visibility into the application's health and warns about performance issues. The `scalemonitor` schema contains information related to this data.

**Example:** Use the following query to check the Disk Space Check Periodicity configuration value.

```
mysql> select * from config where name='APP_KPI_DISK_SPACE_CHECK_INTERVAL_IN_MINUTES';
+----+-----------------------------------------------+-------+-------------------------
-----+
| id | name                                          | value | description
|
+----+-----------------------------------------------+-------+-------------------------
-----+
|  1 | APP_KPI_DISK_SPACE_CHECK_INTERVAL_IN_MINUTES | 60    | Disk space check
periodicity |
+----+-----------------------------------------------+-------+-------------------------
-----+
1 row in set (0.01 sec)
```

**Note:** Although the tables are exposed, they are used internally by the Unified Manager database and have no administrative value to users.

## 3.7 Which Schema to Use

Table 1 shows how to use the schema based on your use case.

**Table 1) Schema mapping information.**

| Schema | Usage |
|---|---|
| netapp_model_view | ▪ Object health and capacity description<br>▪ NAS export policy<br>▪ QoS policy details<br>▪ Snapshots |
| ocum | ▪ Historical capacity data and growth rate for logical and physical storage objects<br>▪ Events<br>▪ Alerts<br>▪ Annotations |
| netapp_performance | ▪ Read Ops, Write Ops, IOPS (min/max), MBPS, logical size (available under sample_qos_volume_workload_* tables), latency<br>▪ Fcpport<br>▪ Aggregate Ops<br>▪ Vserver Ops |
| opm | ▪ Performance threshold information |
| ocum_report | ▪ History view – volumes and aggregates<br>▪ Data protection relationship view |

# 4 netapp_performance Schema Explained

When you construct a performance report, understanding the `performance` schema is a bit complex. Unlike other schema table data, the dynamic nature of the `performance` schema requires some extra effort to understand its structure. This section discusses how to extract performance data from the performance tables.

**Note:** Performance data is collected in 5-minute samples by default, but it is stored differently in the databased based on consumption models.

Some of the tables (other than individual tables based on database objects) have the prefix summary* and some have the prefix sample.*. The following sections explain the meaning of the prefixes.

## 4.1 sample_* Tables

The tables with the prefix sample_* are 5-minute samples that are stored in the Unified Manager database for 30 days.

## 4.2 summary_* Tables

The summaries are hourly statistics in which the 5-minute samples are rolled up into 1-hour samples and stored in the Unified Manager database for up to 13 months.

## 4.3 Understanding the Performance Tables

There are three kinds of tables:

- Tables with the prefix sample_*
- Tables with the prefix summary_*
- Individual tables based on Unified Manager objects

The sample and summary tables help with trending analysis because they are stored for longer periods. These tables are grouped by a unique cluster ID, as recorded by the Unified Manager instance. For example, Table 2 shows how the sample_* tables are categorized, to help you understand how to query the database for performance data.

**Note:** You can follow the same method for tables starting with summary_* prefix.

**Table 2) Understanding the sample_* performance tables.**

| Table Name | Grouped by ID | Information Parameters (Table Description) |
|---|---|---|
| sample_aggregate_* | Cluster ID | Get read, write information on the aggregate |
| sample_cifsvserver | Individual Table | CIFS read, write information |
| sample_cluster | Individual Table | Cluster throughput and utilization by cluster ID |
| sample_disk_* | Cluster ID | Disk-level performance information |
| sample_fcplif | Individual Table | Ops, latency per LIF |
| sample_fcplifvserver | Individual Table | LIF information by LIF ID |
| sample_iscsilif | Individual Table | ISCSI LIC performance information by LIF ID |

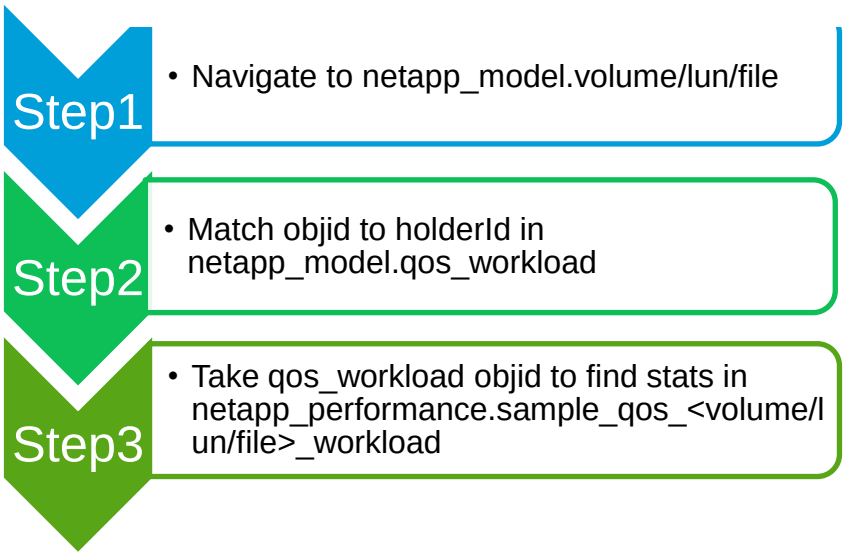
| | | |
|---|---|---|
| sample_lun_* | Cluster ID | Performance information per LUN |
| sample_namespace_* | Cluster ID | Performance information for NVMe namespaces |
| sample_networklif | Individual Table | Throughput and packet information |
| sample_node | Individual Table | Performance information summary per node |
| sample_opm_headroom_aggr_* | Cluster ID | Headroom information |
| sample_qos_lun_workload_* | Cluster ID | QoS information per LUN ID |
| sample_qos_volume_workload_* | Cluster ID | QoS information per volume ID |
| sample_resource_headroom_aggr_* | Cluster ID | Utilization by aggregate ID |

**Note:** You can follow the same method for tables starting with summry_*

## 4.4 Query the QoS Data from the Performance Schema

To query QoS data (for volumes and LUNs) from Unified Manager, you need to pay a bit of attention. The QoS data is dynamic in nature and is stored differently from the other data. To query QoS statistics for ONTAP volumes, LUNs, or files monitored by the Unified Manager instance, follow the three-step procedure in Figure 1.

**Figure 1) Three-step procedure to query QoS data.**




## 4.5 Query the I/O Utilization Data from the Performance Schema

To generate I/O density — that is, IOPS/TB data — from the `netapp_performance` schema for volume and LUN utilization, query Unified Manager by following the steps in the following subsections.

### 4.5.1 For volumes

To find the IOPS/TB of volume 'vol13', follow these steps.

1. Find the volume objid and clusterId in the `netapp_model` schema.

```
mysql> select objid,name,clusterId from netapp_model.volume where
name='vol13';
+-------+-------+-----------+
| objid | name  | clusterId |
+-------+-------+-----------+
|  5506 | vol13 |      4795 |
+-------+-------+-----------+
1 row in set (0.29 sec)
```

2. Find the workload objid in the `netapp_model` schema.

```
mysql> select objid,name,clusterId from netapp_model.qos_workload where
holderId=5506;
+-------+---------------+-----------+
| objid | name          | clusterId |
+-------+---------------+-----------+
|  5864 | vol13-wid16068 |      4795 |
+-------+---------------+-----------+
1 row in set (0.00 sec)
```

3. Get the reference from step 2 and query IOPS/TB from opsPerTB in the `netapp_performance` schema.

```
mysql> select from_unixtime(time/1000), ops, logicalSize, opsPerTb from
netapp_performance.sample_qos_volume_workload_4795 where objid=5864 order by time desc
limit 2;
+--------------------------+-----------+---------------+-----------+
| from_unixtime(time/1000) | ops       | logicalSize   | opsPerTb  |
+--------------------------+-----------+---------------+-----------+
| 2018-05-17 19:09:36.9530 | 0.0833333 | 2050759188480 |  0.044679 |
| 2018-05-17 19:04:36.9530 | 0.0833333 | 2050757386240 | 0.0446791 |
+--------------------------+-----------+---------------+-----------+
```

**Note:** If the volume's logical size (in bytes) is less than 1TB, the opsPerTb column is not calculated and is null. You can find the ops/TB manually with ops/(logicalSize/1024/1024/1024/1024).

### 4.5.2 For LUNs

To find the IOPS/TB of the `'/vol/iscsi_vol1/lun1'` volume, follow these steps.

1. Find the volume objid and clusterId in the `netapp_model` schema.

```
mysql> select objid,path,clusterId from netapp_model.lun where
path='/vol/iscsi_vol1/lun1';
+-------+----------------------+-----------+
| objid | path                 | clusterId |
+-------+----------------------+-----------+
|  2983 | /vol/iscsi_vol1/lun1 |      2437 |
+-------+----------------------+-----------+
1 row in set (1 min 1.99 sec)
```

2. Find the workload objid in the `netapp_model` schema.

```
mysql> select objid,name,clusterId from netapp_model.qos_workload where
holderId=2983;
+-------+---------------+-----------+
| objid | name          | clusterId |
+-------+---------------+-----------+
|  3593 | lun1-wid40794 |      2437 |
+-------+---------------+-----------+
1 row in set (0.08 sec)
```

3. Get the reference from step 2 and query IOPS/TB from opsPerTB in the `netapp_performance` schema.

```
mysql> select from_unixtime(time/1000), ops, logicalSize, opsPerTb from
netapp_performance.sample_qos_lun_workload_2437 where objid=3593 order by time desc
limit 2;
+--------------------------+---------+-------------+----------+
| from_unixtime(time/1000) | ops     | logicalSize | opsPerTb |
+--------------------------+---------+-------------+----------+
| 2018-05-17 19:09:55.2560 | 1138.81 |   133132288 |     NULL |
| 2018-05-17 19:04:55.2550 | 1145.59 |   133132288 |     NULL |
+--------------------------+---------+-------------+----------+
```

**Note:** If the LUN's logical size (in bytes) is less than 1TB, the opsPerTb column is not calculated and is null. You can find the ops/TB manually with ops/(logicalSize/1024/1024/1024/1024).

# 5 Examples

Section 4 described how the exposed schemas can be used to extract the required data. This section looks at some sample SQL constructs that can be used to draw health and capacity data from the Unified Manager instance. As mentioned earlier, you can use any reporting tool that allows a JDBC connection to the Unified Manager MySQL database, such as Microsoft Power BI, Tableau, and MySQL Workbench. But you can choose any reporting tool that is already in use in your enterprise.

## 5.1 Example 1: Find IOPS Utilization per Node (with Customizable Date and Time)

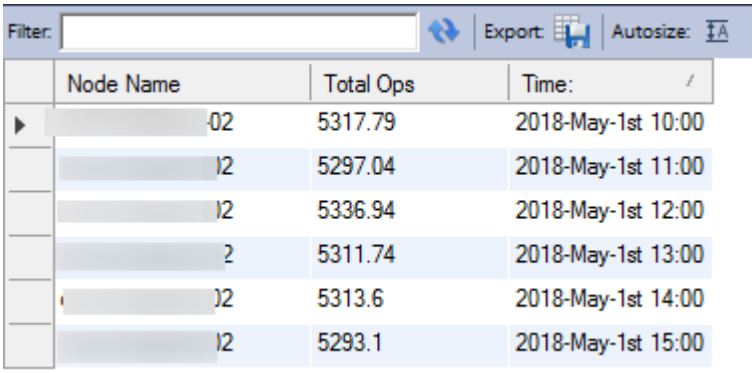
This example finds total ops used per node. You can customize the data based on timestamps.

| Schema Used | Purpose |
|---|---|
| netapp_model_view | Fetch node name and relevant information |
| netapp_performance | Fetch I/O information |

**The Query**

```
SELECT
    netapp_model_view.node.name AS "Node Name",
    netapp_performance.summary_node.totalOps AS "Total Ops",
    FROM_UNIXTIME(netapp_performance.summary_node.fromtime / 1000,
            '%Y-%M-%D %H:%i') AS 'Time:'

FROM netapp_model_view.node

       JOIN

       netapp_performance.summary_node ON netapp_model_view.node.objid =
netapp_performance.summary_node.objid

       AND

       netapp_model_view.node.name = 'ocum-longevity-02'

       WHERE

       FROM_UNIXTIME(netapp_performance.summary_node.fromtime / 1000,
            '%Y-%M-%D %H:%i') BETWEEN '2018-May-1st 10:00' AND '2018-May-1st 15:00';
```

Sample output from a MySQL Workbench tool looks like this.




## 5.2 Example 2: Basic Capacity Report Query for Volumes and Aggregates

This example covers aggregates. Follow the same format for volumes.

| Schema Used | Purpose |
|---|---|
| ocum_report | Fetch historical data for aggregates and volumes |

### The Query

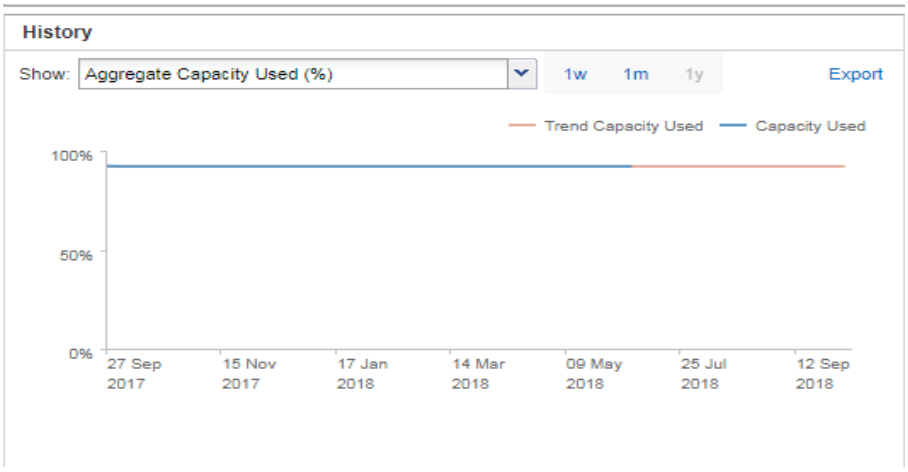
```
SELECT
    ocum_report.aggregate.name AS “Aggregate Name”,
    ocum_report.aggregatecapacityhistoryyearview.periodEndTime AS "Timeframe",
    ocum_report.aggregatecapacityhistoryyearview.UsedSum/1024/1024/1024 AS "Total Used
(GB)"

       FROM
       ocum_report.aggregate
    JOIN
              ocum_report.aggregatecapacityhistoryyearview ON ocum_report.aggregate.id =
ocum_report.aggregatecapacityhistoryyearview.aggregateId

WHERE

       ocum_report.aggregatecapacityhistoryyearview.periodEndTime BETWEEN '2018-05-01'
and '2018-05-30'

       GROUP BY ocum_report.aggregate.name
```

Sample output from a MySQL Workbench tool looks like this.

Filter: Export: Autosize:

| Aggregate Name | Timeframe | Total Used (GB) |
|---|---|---|
| 1 | 2018-05-02 20:00:00 | 1098.079398509115 |
| 2 | 2018-05-02 20:00:00 | 902.145597819239 |
| 0 | 2018-05-02 20:00:00 | 454.308865800500 |
| | 2018-05-02 20:00:00 | 268.943466108292 |
| r0 | 2018-05-02 20:00:00 | 267.205590665340 |
| 0 | 2018-05-02 20:00:00 | 142.958444684744 |
| 0 | 2018-05-02 20:00:00 | 93.752863269299 |
| | 2018-05-02 20:00:00 | 60.155718792230 |

**Note:** Capacity trending charts are also available in the Unified Manager GUI. You can select a time range of 1 week, 1 month, or 1 year. In the GUI, navigate to Health → Aggregates (or) Volumes. Click a single object (aggregate or volume) and find the details on the object summary page, as shown in the following screenshot.



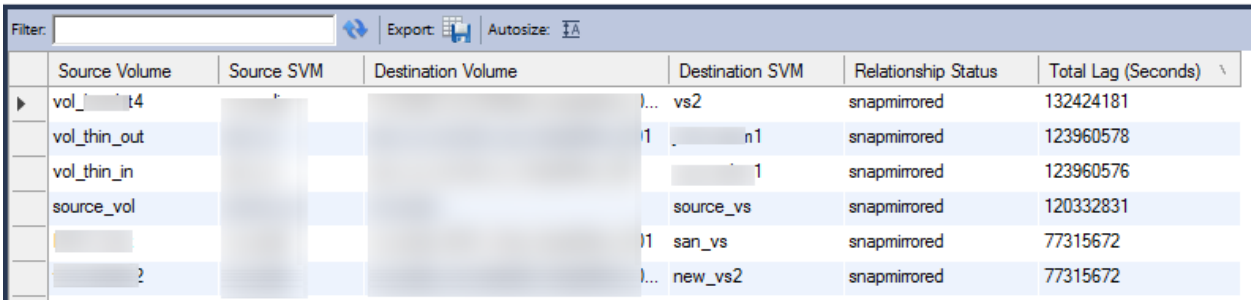
## 5.3 Example 3: Protection Relationships Showing Relationship Status and Lag Time

These reports are available in the GUI itself. However, if you want to create a SQL query for generating reports with your own reporting tool, use the following example.

### The Query

```
SELECT

       sourceVolumeName AS "Source Volume",
       sourceVserverName AS "Source SVM",
       destinationVolumeName AS "Destination Volume",
       destinationVserverName AS "Destination SVM",
       relationshipState AS "Relationship Status",
       lagDuration AS "Total Lag (Seconds)"

FROM

    ocum_report.volumerelationships;
```

Sample output from a MySQL Workbench tool looks like this.




| Source Volume | Source SVM | Destination Volume | Destination SVM | Relationship Status | Total Lag (Seconds) |
|---|---|---|---|---|---|
| vol_ t4 | | | vs2 | snapmirrored | 132424181 |
| vol_thin_out | | | n1 | snapmirrored | 123960578 |
| vol_thin_in | | | 1 | snapmirrored | 123960576 |
| source_vol | | | source_vs | snapmirrored | 120332831 |
| | | | san_vs | snapmirrored | 77315672 |
| | | | new_vs2 | snapmirrored | 77315672 |

**Note:** There are many fields and counters to choose from. A truncated output of the contents of the ocum_report.volumerelationships table looks like this.

```
mysql> describe ocum_report.volumerelationships;
+-----------------------------------+----------------
| Field                             | Type
+-----------------------------------+----------------
| id                                | bigint(20)
| controlPlane                      | varchar(255)
| relationshipType                  | varchar(255)
| relationshipState                 | varchar(255)
| isHealthy                         | tinyint(1)
| lastTransferDuration              | bigint(20)
| lastTransferSize                  | bigint(20)
| lastTransferRate                  | decimal(23,4)
| lastTransferStartTime             | bigint(21)
| lastTransferEndTime               | bigint(11)
| lagDuration                       | bigint(20)
| relationshipStatus                | varchar(255)
| name                              | varchar(2048)
| healthStatus                      | int(11)
| overallStatus                     | enum('NORMAL','
| lagStatus                         | enum('NORMAL','
| lastSuccessfulUpdateTime          | bigint(11)
| destinationPath                   | text
| destinationVolumeId               | bigint(20)
| destinationVolumeName             | varchar(255)
| destinationVolumeState            | varchar(255)
| destinationVolumeType             | varchar(255)
| destinationVolumeSize             | bigint(20)
| destinationVserverId              | bigint(20)
| destinationVserverName            | varchar(255)
| destinationAggregateId            | bigint(20)
| destinationAggregateName          | varchar(255)
| destinationClusterNodeId          | bigint(20)
| destinationClusterNodeName        | varchar(255)
| destinationClusterId              | bigint(20)
| destinationClusterName            | varchar(255)
| destinationClusterVersion         | varchar(255)
| destinationClusterVersionGeneration | int(11)
| destinationClusterVersionMajor    | int(11)
| destinationClusterVersionMinor    | int(11)
| sourcePath                        | text
```

## 5.4 Example 4: Query Your QoS Policy

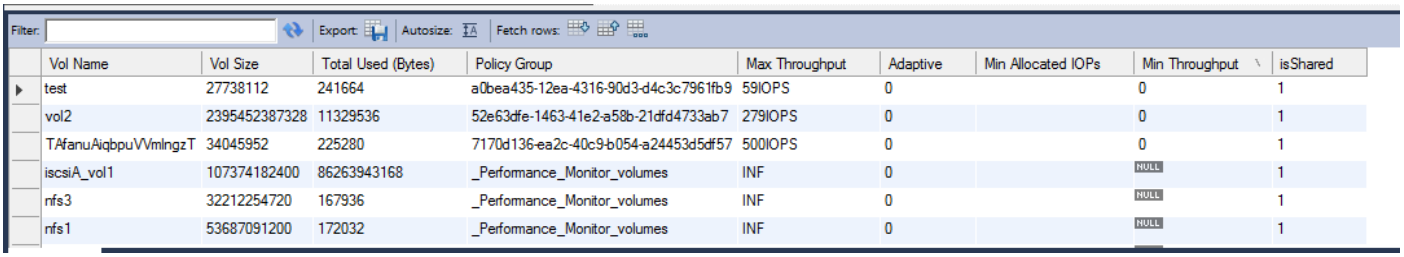
Here is a sample query that you can use to get QoS policy group details for all the volume objects monitored by your Unified Manager instance.

| Schema Used | Purpose |
|---|---|
| netapp_model_view | Fetch QoS policy group and other volume information |

**The Query**

```
SELECT

        v.name AS "Vol Name",
        v.size AS "Vol Size",
        v.sizeUsed AS "Total Used (Bytes)",
        q.policyGroup AS "Policy Group",
        q.maxThroughput AS "Max Throughput",
        q.isAdaptive "Adaptive",
        q.minIOPsAllocation "Min Allocated IOPs",
        q.minThroughput "Min Throughput",
        q.isShared

FROM

        netapp_model_view.volume v,
        netapp_model_view.qos_policy_group q

WHERE

        v.qosPolicyGroupId IS NOT NULL

        AND

        v.state = 'ONLINE'

        AND

        q.objid = v.qosPolicyGroupId
```

Sample output from a MySQL Workbench tool looks like this.



| Vol Name | Vol Size | Total Used (Bytes) | Policy Group | Max Throughput | Adaptive | Min Allocated IOPs | Min Throughput | isShared |
|---|---|---|---|---|---|---|---|---|
| test | 27738112 | 241664 | a0bea435-12ea-4316-90d3-d4c3c7961fb9 | 59IOPS | 0 | | 0 | 1 |
| vol2 | 2395452387328 | 11329536 | 52e63dfe-1463-41e2-a58b-21dfd4733ab7 | 279IOPS | 0 | | 0 | 1 |
| TAfanuAiqbpuVVmlngzT | 34045952 | 225280 | 7170d136-ea2c-40c9-b054-a24453d5df57 | 500IOPS | 0 | | 0 | 1 |
| iscsiA_vol1 | 107374182400 | 86263943168 | _Performance_Monitor_volumes | INF | 0 | | NULL | 1 |
| nfs3 | 32212254720 | 167936 | _Performance_Monitor_volumes | INF | 0 | | NULL | 1 |
| nfs1 | 53687091200 | 172032 | _Performance_Monitor_volumes | INF | 0 | | NULL | 1 |

# 6 Resources and Information

## 6.1 Database Schema Documentation

Beginning with version 7.3, NetApp has opened the Unified Manager database so that external reporting tools can extract processed data from Unified Manager over a JDBC connection. For detailed documentation of the database schema, refer to the following guides.

- Unified Manager 7.3 schema documentation:

https://library.netapp.com/ecm/ecm_download_file/ECMLP2846645

- Unified Manager 9.4 schema documentation:

https://library.netapp.com/ecm/ecm_download_file/ECMLP2846163

### 6.2 Other Resources

- Creating operational custom reports with Unified Manager (technical report):

https://www.netapp.com/us/media/tr-4565.pdf

- Unified Manager resource page:

https://mysupport.netapp.com/info/web/ECMLP2688087.html

- Unified Manager blogs and podcasts:

https://blog.netapp.com/?s=unified+manager

## Version History

| Version | Date | Author | Description |
|---|---|---|---|
| 1.0 | August 2018 | Dhiman Chakraborty | Unified Manager schema definitions, outline, and usage with examples. |

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.