Unleash the Value of the Data in Your OEM Repository

Iordan K. Iotzov

iiotzov@newsamerica.com
News America Marketing
(NewsCorp)
Sources of Information in OEM

**AWR/ASH**
- SYSAUX
  - DB1
- SYSAUX
  - DB2
- SYSAUX
  - DB3

**OEM Repository**
- OEM DB
Process of Metrics Gathering in OEM

Databases
- OS
- Upload files
- OEM agent

OEM Server

OEM Repository
- MGMT$METRIC_CURRENT
- MGMT$METRIC_DETAILS
- MGMT$METRIC_HOURLY
- MGMT$METRIC_DAILY
Useful Applications – averages over any period of time

Find the average run queue length during business hours (M-F, 7 am – 8 pm) for 30 days
Useful Applications – averages over any period of time

Find the average run queue length during business hours (M-F, 7 am – 8 pm) for 30 days

Query:

```
select avg(average) 
from mgmt$metric_hourly
where rollup_timestamp > sysdate – 30 
  and target_name = 'dbtest01'
  and metric_name = 'Load'
  and column_label = 'Run Queue Length (1 minute average)'
  and to_char(rollup_timestamp,'DAY') not in ('SATURDAY','SUNDAY')
  and rollup_timestamp between trunc(rollup_timestamp,'DD') + 7/24 
    and trunc(rollup_timestamp,'DD') + 20/24
```

Output: 2.97053285256410256410256410256410256411

Opportunities: Unlimited flexibility in managing OEM metrics data
Useful Applications – enforcing enterprise-wide policies

Monitor Force Logging mode for all production databases.

Create UDM in each production database. The sole purpose of this UDM would be to supply data to the OEM repository.
Useful Applications – enforcing enterprise-wide policies

Monitor Force Logging mode for all production databases.
Create UDM in the OEM repository database. This UDM will notify/page DBAs if a DB is not in Force Logging mode

```
create UDM in the OEM repository database. This UDM will notify/page DBAs if a DB is not in Force Logging mode
```
Useful Applications – enforcing enterprise-wide policies

Monitor Force Logging mode for all production databases.

The full query behind the OEM repository UDM:

```sql
select count( member_target_guid )
from mgmt$group_derived_memberships o , mgmt$target t
where o.composite_target_name = 'PROD'
and o. member_target_guid = t.target_guid
and ( t.target_type = 'rac_database'
or (t.target_type = 'oracle_database'
and t.type_qualifier3 != 'RACINST'
) )
and not exists ( select *
from mgmt$metric_current i
where i.target_guid = o.member_target_guid
and metric_name = 'SQLUDM'
and column_label = 'ForcedLogging'
and Metric_Column = 'StrValue'
and collection_timestamp > sysdate - 20/1440
and value = 'YES'
)
Useful Applications – enforcing enterprise-wide policies

Monitor Force Logging mode for all production databases.

How about exempting a database? One need to do a concerted effort to exempt a database.
Useful Applications – enforcing enterprise-wide policies

Monitor Force Logging mode for all production databases.

Ability to deliver customized solutions – all but two tablespaces (TBLS_STG, TBLS_UNRECOVERABLE) must be in Force Logging mode
Useful Applications – integration with other data sources

Combining ASH/AWR and OEM repository

Where does the variation in single block read time come from?

```
select corr(db.time_waited, oem.value), count(*)
from (select sample_time, time_waited
       from dba_hist_active_sess_history@prod_db
       where event = 'db file sequential read'
       and session_state = 'WAITING'
       and sample_time > sysdate - 1 ) db,
       ( select collection_timestamp, value
         from mgmt$metric_details
         where target_name = '+ASM_PROD'
         and metric_name = 'Single_Instance_DiskGroup_Performance'
         and metric_column = '<ASM Metric>' ) oem
where oem.collection_timestamp between db.sample_time - 15/1440
  and db.sample_time

<table>
<thead>
<tr>
<th>ASM Metric</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>writes_ps</td>
<td>0.04</td>
</tr>
<tr>
<td>reads_ps</td>
<td>-0.04</td>
</tr>
<tr>
<td>write_throughput</td>
<td>0.04</td>
</tr>
<tr>
<td>read_throughput</td>
<td>0.01</td>
</tr>
</tbody>
</table>
```

Very different sampling rates usually result in lower correlation
Useful Applications – the power of Oracle’s own SQL

Why is the latency of Streams Apply so high? Could it be the disk IO? It is often the issue...

Check: 
```
select 
corr(streams_latency.value, IO_load.value)
from
( select * 
  from 
  mgmt$metric_details 
  where 
  target_name = 'STRDEST'
  and 
  metric_name = 'streams_latency_throughput'
  and 
  column_label = 'Latency' ) streams_latency , 
( select * 
  from 
  mgmt$metric_details 
  where 
  target_name = 'STRDEST'
  and 
  metric_name = 'instance_throughput'
  and 
  column_label = 'I/O Megabytes (per second)' ) IO_load
where 
  streams_latency.collection_timestamp between IO_load.collection_timestamp 
  and IO_load.collection_timestamp + 5/(60*24)
and 
  streams_latency.collection_timestamp > sysdate - 1
```

Output:  -0.001043436634863007975354458994748444098631

Interpretation: Hypothesis rejected! Need to look for other explanation...
Useful Applications – the power of Oracle’s own SQL

Why is the latency of Streams Apply so high? Maybe it is the redo volume on the Capture side...

**Check:**

```sql
select corr(streams_latency.value, redo_source_1.value + redo_source_2.value)
from
    ( select *
      from mgmt$metric_details
      where target_name = 'STRDEST'
      and metric_name = 'streams_latency_throughput'
      and column_label = 'Latency' ) streams_latency,
    (select *
      from mgmt$metric_details
      where target_name = 'STRSRC_STRSRC 1'
      and metric_name = 'instance_throughput'
      and column_label = 'Redo Generated (per second)' ) redo_source_1,
    (select *
      from mgmt$metric_details
      where target_name = 'STRSRC_STRSRC 2'
      and metric_name = 'instance_throughput'
      and column_label = 'Redo Generated (per second)' ) redo_source_2
where
    streams_latency.collection_timestamp between redo_source_1.collection_timestamp
    and redo_source_1.collection_timestamp + 5/(60*24)
and
    streams_latency.collection_timestamp between redo_source_2.collection_timestamp
    and redo_source_2.collection_timestamp + 5/(60*24)
and
    streams_latency.collection_timestamp > sysdate – 1
```

**Output:** 0.7641297491294021289799243642319279128629

**Interpretation:** The redo volume on the Capture side is likely the reason for increased latency on the Apply side...
Advanced Forecasting Example - tablespace size forecast

Simple linear regression:

\[ \text{tbls}(t) = \text{tbls}_0 + \text{incr} \cdot t \]

- \( \text{tbls}_0 \) (Intercept) – tablespace size for time 0
- \( \text{incr} \) (Slope) – how fast a tablespace grows
- \( t \) – time
- \( \text{tbls}(t) \) – tablespace size at time \( t \)
Advanced Forecasting Example - tablespace size forecast

Quality of forecasting - 95% prediction interval of the regression line
Advanced Forecasting Example - tablespace size forecast

Getting raw historical tablespace sizing data from the OEM repository

```
insert into raw_data
select
    m.rollup_timestamp ,
    m.average
from
    mgmt$metric_daily m ,
    mgmt$target_type t
where
    (t.target_type = 'rac_database'
    or (t.target_type = 'oracle_database'
        and t.type_qualifiers3 != 'RACINST'))
and
    m.target_guid = p_target_guid (Database)
and
    m.target_guid = t.target_guid
and
    m.metric_guid = t.metric_guid
and
    t.metric_name = 'tbspAllocation'
and
    t.metric_column = 'spaceUsed'
and
    m.rollup_timestamp >= sysdate - p_period_hist
and
    m.rollup_timestamp <= sysdate
and
    m.key_value = p_tablespace_name;
```
Advanced Forecasting Example - tablespace size forecast

Utilize Oracle’s built-in packages and functions to compute most of the statistics in this project

```sql
select 
  regr_intercept (average , rollup_timestamp - sysdate ) ,
  regr_slope (average , rollup_timestamp - sysdate ) ,
  regr_r2 (average , rollup_timestamp - sysdate ) ,
  regr_count (average , rollup_timestamp - sysdate ) ,
  regr_avgx (average , rollup_timestamp - sysdate ) ,
  regr_sxx (average , rollup_timestamp - sysdate ) ,
  regr_syy (average , rollup_timestamp - sysdate ) ,
  regr_sxy (average , rollup_timestamp - sysdate )
into 
  ..... 
from 
  raw_data
```

Import data and create functions that are not available in Oracle

```sql
create or replace
FUNCTION f_forecast_t_dist (p_degree_freedom NUMBER , p_conf NUMBER) RETURN NUMBER IS
  l_res NUMBER;
  l_one_tail_conf NUMBER;
BEGIN
  --- Custom Code
END f_forecast_t_dist;
```
Advanced Forecasting Example - tablespace size forecast

Final computations

```plaintext
l_ci:=sqrt ( 1/ l_cnt
    +
    power((p_period_forecast - l_avgx),2)
    /
    l_sxx
);

l_sigma:= sqrt( abs(
   (l_syy
   -
   (power(l_sxy,2)/l_sxx)
)
   /(l_cnt-2)
))
;

p_out_estimate_95_range:=(l_ci
*  
  f_forecast_t_dist(l_cnt-2,0.95)
*  
  l_sigma)
;
```
**Advanced Forecasting Example - tablesapce size forecast**

**Delivery Methods - Various**

---

**Edit Report Definition:** "Production Oracle Database Tablespaces in need of sizing review"

**Set Parameters**

- **Element Type:** Table from SQL
- **Applicable Target Types:** Any Target
- **Element Description:** Displays query results as a table

**Options**

- **Statement:**
  ```sql
  select forecast_type "Forecast Parameters", db_name "Database Name", tablespace_name "Tablespace Name", current_size "Current Size", estimate_value "Forecasted Size", +/- estimate_95_band "Range for Forecasted Size", +/- estimate_95_band_cdf "CDF for Forecasted Size", estimate_normality "Estimate of Normality", estimate_autocorr "Estimate of Autocorrelation", estimate_heteroscedasticity "Estimate of Heteroscedasticity", estimate_r2 "Estimate of Determination" from (SELECT * from table(FORECAST_EST_PROD_DBS (90,45,90,90,FIT)) A WHERE a.TABLESPACE_NAME = 'TBL0')
  union
  SELECT * from table(FORECAST_EST_PROD_DBS (90,90,180,FIT)) B
  WHERE B.TABLESPACE_NAME <> 'TBL0'
  |

- **Statement Type:** SQL
- **Rows to Display in Table:** 990
### Advanced Forecasting Example - tablespace size forecast

**Sample Output:**

#### Tablespace Growth Fits Linear Regression Model

<table>
<thead>
<tr>
<th>Forecast Parameters</th>
<th>Database Name</th>
<th>Tablespace Name</th>
<th>Current Size</th>
<th>Forecasted Size</th>
<th>Range for Forecasted Size</th>
<th>Range for Forecasted Size (%)</th>
<th>Suggested Size</th>
<th>Estimate of Normality</th>
<th>Estimate of Autocorrelation</th>
<th>Estimate of Heteroscedasticity</th>
<th>Estimate of Determination</th>
</tr>
</thead>
<tbody>
<tr>
<td>90 days look back - 90 days forecast - 180 days sizing goal</td>
<td>DRRP</td>
<td>DRRP_DATA</td>
<td>83,100</td>
<td>98,199</td>
<td>+/- 26654</td>
<td>+/- 27%</td>
<td>128,354</td>
<td>0</td>
<td>0.65</td>
<td>0.27</td>
<td>0.54</td>
</tr>
<tr>
<td>90 days look back - 90 days forecast - 180 days sizing goal</td>
<td>ZFFO</td>
<td>LFD_DATA</td>
<td>16,000</td>
<td>17,645</td>
<td>+/- 3182</td>
<td>+/- 18%</td>
<td>25,518</td>
<td>0.83</td>
<td>0.56</td>
<td>0.06</td>
<td>0.87</td>
</tr>
<tr>
<td>90 days look back - 90 days forecast - 180 days sizing goal</td>
<td>ZFFO</td>
<td>LFD_INDEX</td>
<td>3,100</td>
<td>3,589</td>
<td>+/- 783</td>
<td>+/- 21%</td>
<td>5,282</td>
<td>0.87</td>
<td>0.41</td>
<td>0.01</td>
<td>0.83</td>
</tr>
<tr>
<td>90 days look back - 90 days forecast - 180 days sizing goal</td>
<td>ZFFO</td>
<td>POSL_DATA</td>
<td>344,177</td>
<td>610,716</td>
<td>+/- 77604</td>
<td>+/- 12%</td>
<td>903,460</td>
<td>0.21</td>
<td>0.54</td>
<td>0.23</td>
<td>0.94</td>
</tr>
<tr>
<td>90 days look back - 90 days forecast - 180 days sizing goal</td>
<td>ZFFO</td>
<td>DMWV_DATA</td>
<td>19,000</td>
<td>20,224</td>
<td>+/- 3147</td>
<td>+/- 15%</td>
<td>25,342</td>
<td>0.26</td>
<td>0.71</td>
<td>0.28</td>
<td>0.74</td>
</tr>
<tr>
<td>90 days look back - 90 days forecast - 180 days sizing goal</td>
<td>ZFFO</td>
<td>SYSTEM</td>
<td>20,520</td>
<td>34,678</td>
<td>+/- 14119</td>
<td>+/- 40%</td>
<td>54,144</td>
<td>0.23</td>
<td>0.74</td>
<td>0.08</td>
<td>0.67</td>
</tr>
<tr>
<td>90 days look back - 90 days forecast - 180 days sizing goal</td>
<td>WPPPT</td>
<td>TXCD</td>
<td>150</td>
<td>153</td>
<td>+/- 33</td>
<td>+/- 21%</td>
<td>197</td>
<td>0.55</td>
<td>0.54</td>
<td>0.08</td>
<td>0.64</td>
</tr>
</tbody>
</table>

#### Tablespace Growth Does not Fit Linear Regression Model

<table>
<thead>
<tr>
<th>Forecast Parameters</th>
<th>Database Name</th>
<th>Tablespace Name</th>
<th>Current Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>90 days look back - 45 days forecast - 180 days sizing goal</td>
<td>LSPPR</td>
<td>TEMP</td>
<td>1,000</td>
</tr>
<tr>
<td>90 days look back - 45 days forecast - 180 days sizing goal</td>
<td>LSPPR</td>
<td>UNDOTBS1</td>
<td>516</td>
</tr>
<tr>
<td>90 days look back - 45 days forecast - 180 days sizing goal</td>
<td>JBOP</td>
<td>TEMP</td>
<td>58</td>
</tr>
<tr>
<td>90 days look back - 45 days forecast - 180 days sizing goal</td>
<td>FQQ</td>
<td>TEMP</td>
<td>14,300</td>
</tr>
</tbody>
</table>
References

➢ *Metalink Note 831243.1 (Examples: Creating Custom Reports).*

➢ *Oracle® Enterprise Manager Extensibility 10g Release 2 (10.2) for Windows or UNIX*


OEM 11g appears to have the same OEM repository structures as OEM 10g