ORACLE
FUSION MIDDLEWARE
BUSINESS ACTIVITY
MONITORING

# ACTIVE DASHBOARDS IN ORACLE BAM 12.1.3

ORACLE WHITE PAPER | SEPTEMBER 2014

ORACLE

## Disclaimer

## Table of Contents

## Introduction

Active Data Service (ADS) enhances your analytic experience with live data by allowing Oracle BAM to feed real time data to dashboards and Key Performance Indicator (KPI) watchlists as and when the data arrives into the Oracle BAM system. This enables you to take decisions early on into a process by quickly visualizing trend detections, which may lead to SLA violations or prove to be critical in other ways.

## ADS Scenario:

Here is a scenario to help understand why you may need real time updates. Consider a Film Sales company that wants to monitor sales activity across all sales regions, sales trends, total sales, and so on, as and when sales happen. This ability to monitor business events in real time, helps you take informed decisions and appropriate inventory management actions by understanding sales trends across brands, categories, regions, and states. Using BAM 12c, you can create the right Data Objects with fields like Sales, Region, State, Brand, and Category as columns to create an appropriate business query through BAM Composer. For instance, the business query can be designed to capture the sum of sales data organized by region. The business query in this case will be:

“SELECT REGION AS REGION, SUM(SALES) AS SUMSALES FROM BEAM_VIEW_46 GROUP BY REGION ORDER BY REGION ASC”.

Here, REGION is the dimension that you use to group data. SALES is the fact data using which the SUM aggregation is calculated. BEAM_VIEW_46 represents the FilmSales DataObject in BAM. You can also choose to order the data by the REGION. This example represents the default form of the query. To turn this into a real time query, you can turn it into an ‘Active’ query in Oracle BAM using either the Business View composer or when you open the dashboard. As you do this, a corresponding continuous query is registered into the CQService.

## Understanding Active Viewsets

An active viewset is a server-side entity which captures a business query along with its various filters and parameters, and its results. ViewSet is the sharing between clients who look at the same business view. ViewSet exists only for active views. It is responsible for executing a query with various downstream components like CQService via the common query module. An Active ViewSet caches this result in a distributed cache and is accessible from any BAM server in the cluster. This is tagged as the master viewset. If the same business view is opened by another user, it is still represented by an Active ViewSet but is tagged as a ‘slave viewset’ of the corresponding master viewset. This avoids running the query again when the same business view is opened multiple times. When the last business view is closed in the browser, this viewset is closed as well. It also expires if the dashboard loses its connectivity with the server for more than three minutes.

## Understanding Queries

Queries can be active or static. Oracle BAM uses certain query types to help retrieve data for various visualizations available within dashboards. These query types are detailed as follows.

1. **Continuous Queries**: The active viewset registers a continuous query with CQService. CQService is an Oracle BAM component which interacts with the CQL engine and is responsible for delivering continuous query results. Here is an example of the registered continuous query:

   'CREATE QUERY FilmSalesQuery_V1_0_SN7 as SELECT REGION AS REGION, SUM(SALES) AS SUMSALES FROM FilmSales GROUP BY REGION EVALUATE EVERY 1 SECONDS PRIMARY KEY ( REGION) destination "jms:queue/oracle.beam.cqservice.mdbs.reportcache:queuecf/oracle.beam.cqservice.mdbs.reportcache?batch=true,semantic=update"'.

   This query enables you to receive the change data streams from CQService into the active viewset via JMS on the jms:queue/oracle.beam.cqservice.mdbs.reportcache queue. The EVALUATE clause helps in throttling the active data push into the ActiveViewset from CQService. Here, data is pushed into the active viewset every second. The 'Evaluate' feature can be turned on when you turn this query to a continuous query. See more in the Enabling the Dashboards section.

2. **Group Queries**: Group queries are created when you need data from various attributes to come in when you type a single query. A static dashboard based on group queries can be converted to an active dashboard. For example, you can create a dashboard based on a group query "sum(sales) group by region". Once this is converted to an active dashboard, you can visualize the dimension data grouped by a dimension field and aggregate measures like "sum of sales" data changing over time, pulled into the dashboard. You will see the pie resizing itself as and when its data changes.
   You can also see the bar chart based on the group query "sum(sales) group by region"

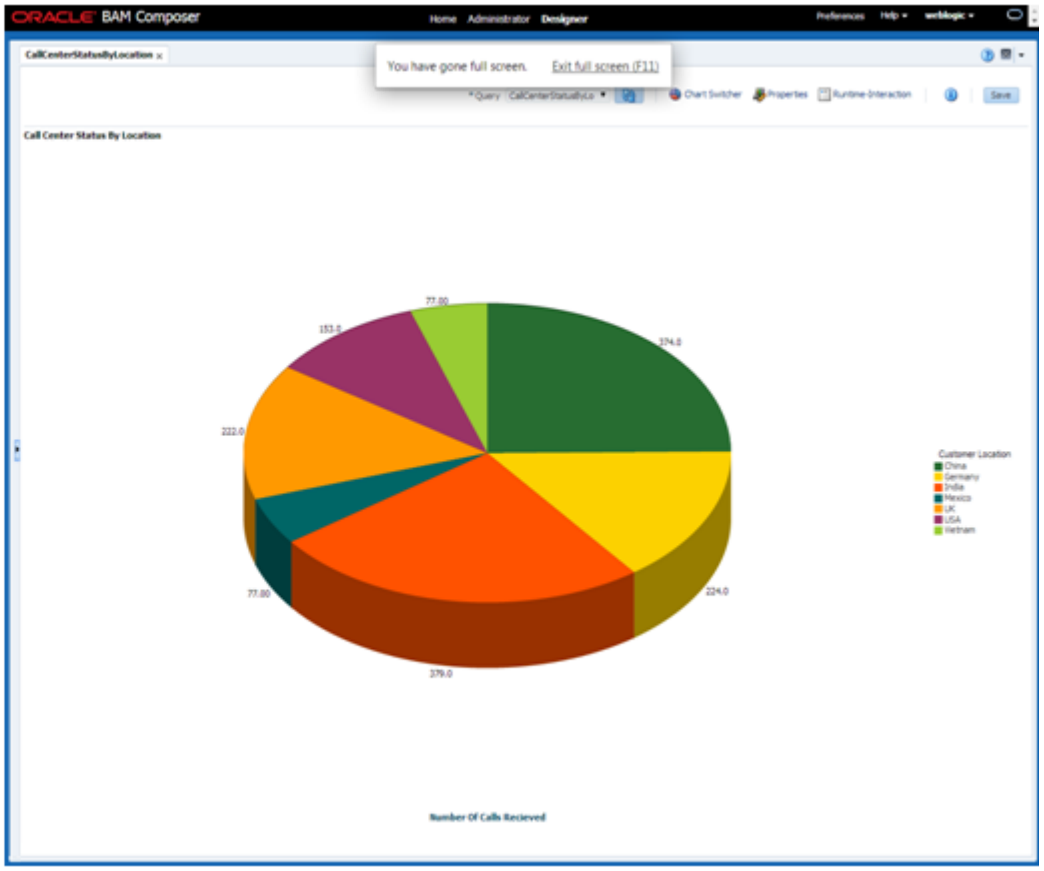

ORACLE BAM Composer
Home Administrator Designer
Preferences Help weblogic
CallCenterStatusByLocation
You have gone full screen. Exit full screen (F11)
Query CallCenterStatusByLo
Chart Switcher Properties Runtime Interaction Save
Call Center Status By Location
77.00
374.0
153.0
222.0
224.0
77.00
379.0
Customer Location
China
Germany
India
Mexico
UK
USA
Vietnam
Number Of Calls Recieved

Figure 1 – Pie Chart Showing Sales by Location

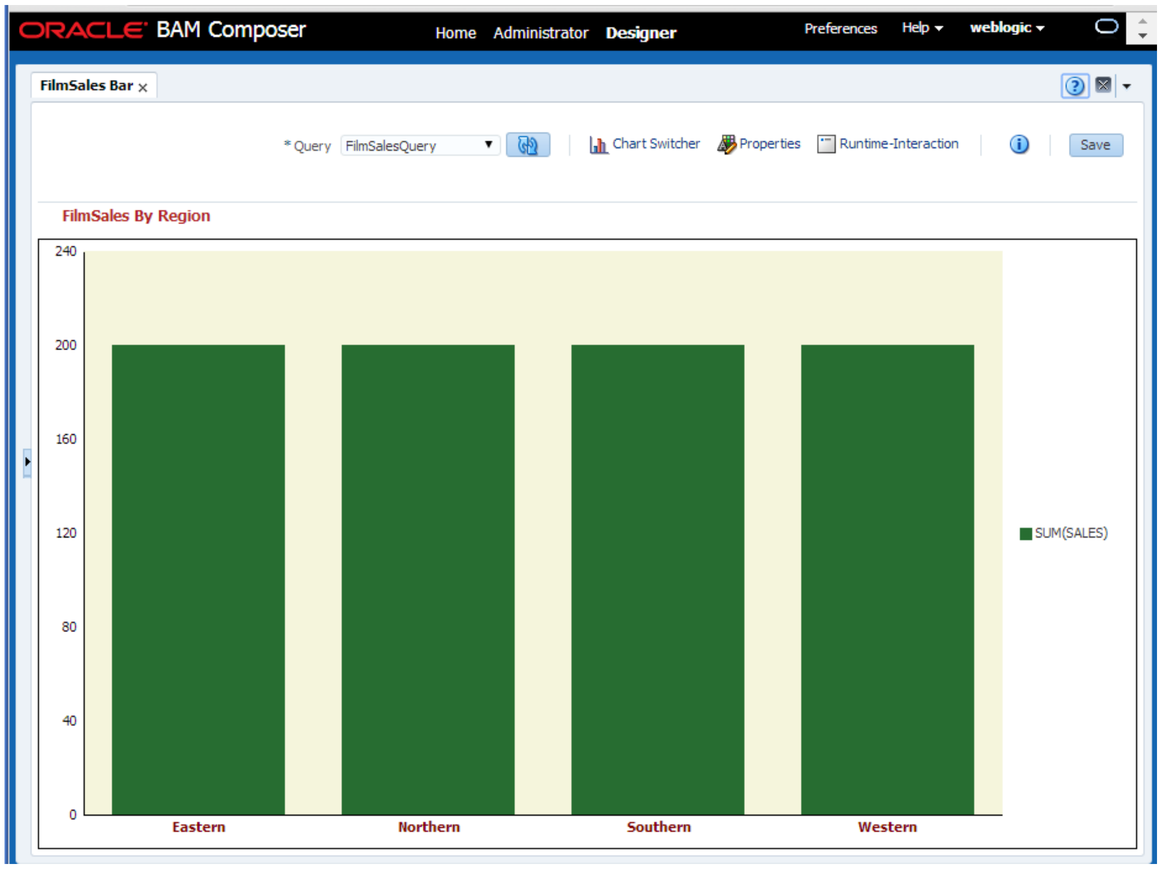
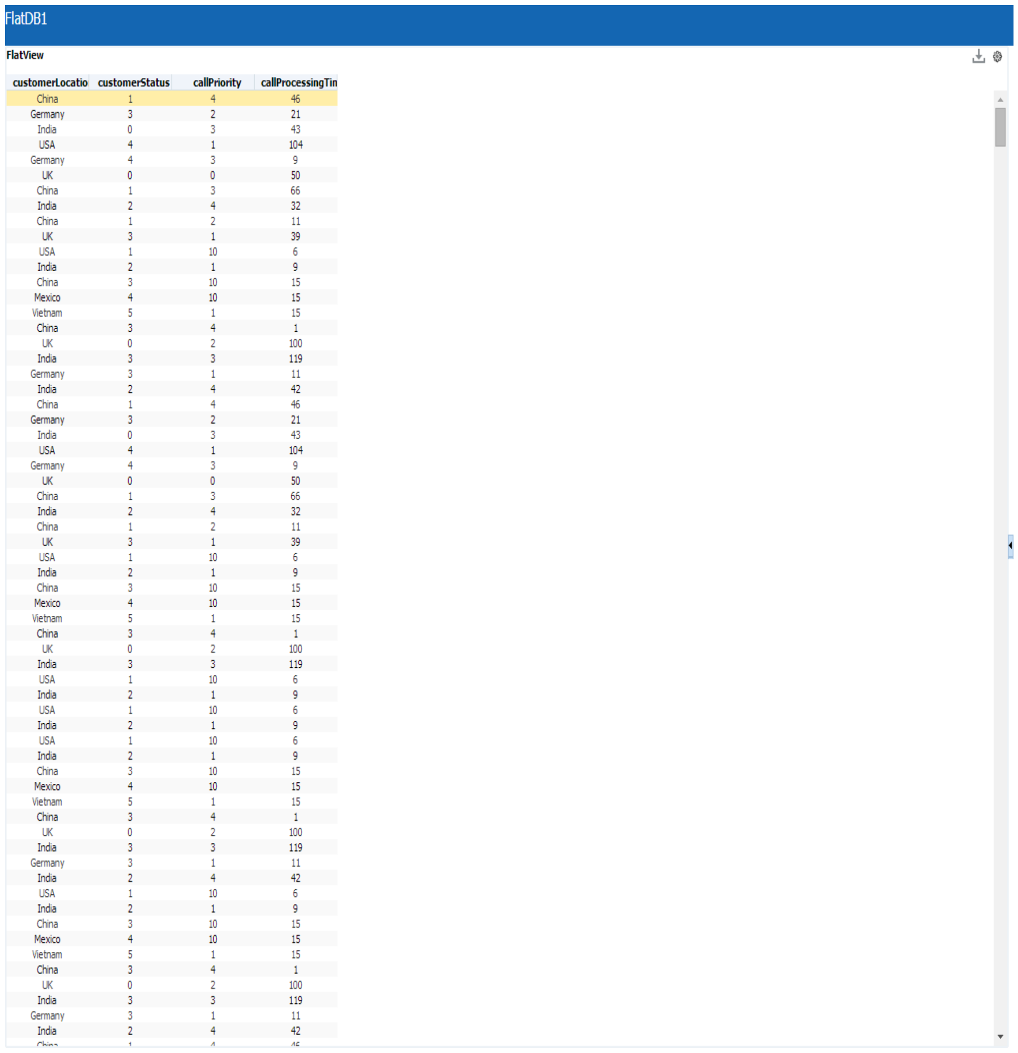


Figure 2 – Bar chart showing sales by location

3. **Flat Queries**: Flat query-based dashboards bound to a table view can be converted to an active dashboard. For example, "select call_duration, region from call_center". Here, if a matching row is added or updated in the DataObject, that row shows up on the top of the dashboard.


FlatDB1

FlatView

| customerLocatio | customerStatus | callPriority | callProcessingTin |
|---|---|---|---|
| China | 1 | 4 | 46 |
| Germany | 3 | 2 | 21 |
| India | 0 | 3 | 43 |
| USA | 4 | 1 | 104 |
| Germany | 4 | 3 | 9 |
| UK | 0 | 0 | 50 |
| China | 1 | 3 | 66 |
| India | 2 | 4 | 32 |
| China | 1 | 2 | 11 |
| UK | 3 | 1 | 39 |
| USA | 1 | 10 | 6 |
| India | 2 | 1 | 9 |
| China | 3 | 10 | 15 |
| Mexico | 4 | 10 | 15 |
| Vietnam | 5 | 1 | 15 |
| China | 3 | 4 | 1 |
| UK | 0 | 2 | 100 |
| India | 3 | 3 | 119 |
| Germany | 3 | 1 | 11 |
| India | 2 | 4 | 42 |
| China | 1 | 4 | 46 |
| Germany | 3 | 2 | 21 |
| India | 0 | 3 | 43 |
| USA | 4 | 1 | 104 |
| Germany | 4 | 3 | 9 |
| UK | 0 | 0 | 50 |
| China | 1 | 3 | 66 |
| India | 2 | 4 | 32 |
| China | 1 | 2 | 11 |
| UK | 3 | 1 | 39 |
| USA | 1 | 10 | 6 |
| India | 2 | 1 | 9 |
| China | 3 | 10 | 15 |
| Mexico | 4 | 10 | 15 |
| Vietnam | 5 | 1 | 15 |
| China | 3 | 4 | 1 |
| UK | 0 | 2 | 100 |
| India | 3 | 3 | 119 |
| USA | 1 | 10 | 6 |
| India | 2 | 1 | 9 |
| USA | 1 | 10 | 6 |
| India | 2 | 1 | 9 |
| USA | 1 | 10 | 6 |
| India | 2 | 1 | 9 |
| China | 3 | 10 | 15 |
| Mexico | 4 | 10 | 15 |
| Vietnam | 5 | 1 | 15 |
| China | 3 | 4 | 1 |
| UK | 0 | 2 | 100 |
| India | 3 | 3 | 119 |
| Germany | 3 | 1 | 11 |
| India | 2 | 4 | 42 |
| USA | 1 | 10 | 6 |
| India | 2 | 1 | 9 |
| China | 3 | 10 | 15 |
| Mexico | 4 | 10 | 15 |
| Vietnam | 5 | 1 | 15 |
| China | 3 | 4 | 1 |
| UK | 0 | 2 | 100 |
| India | 3 | 3 | 119 |
| Germany | 3 | 1 | 11 |
| India | 2 | 4 | 42 |

Figure 3 – Flat Query View

4. **Aggregate queries without dimensions**: Aggregate queries are defined at the DataObject level. For example, “avg(sales) from FilmSales” are used by gauge-based dashboards, and they support active data.

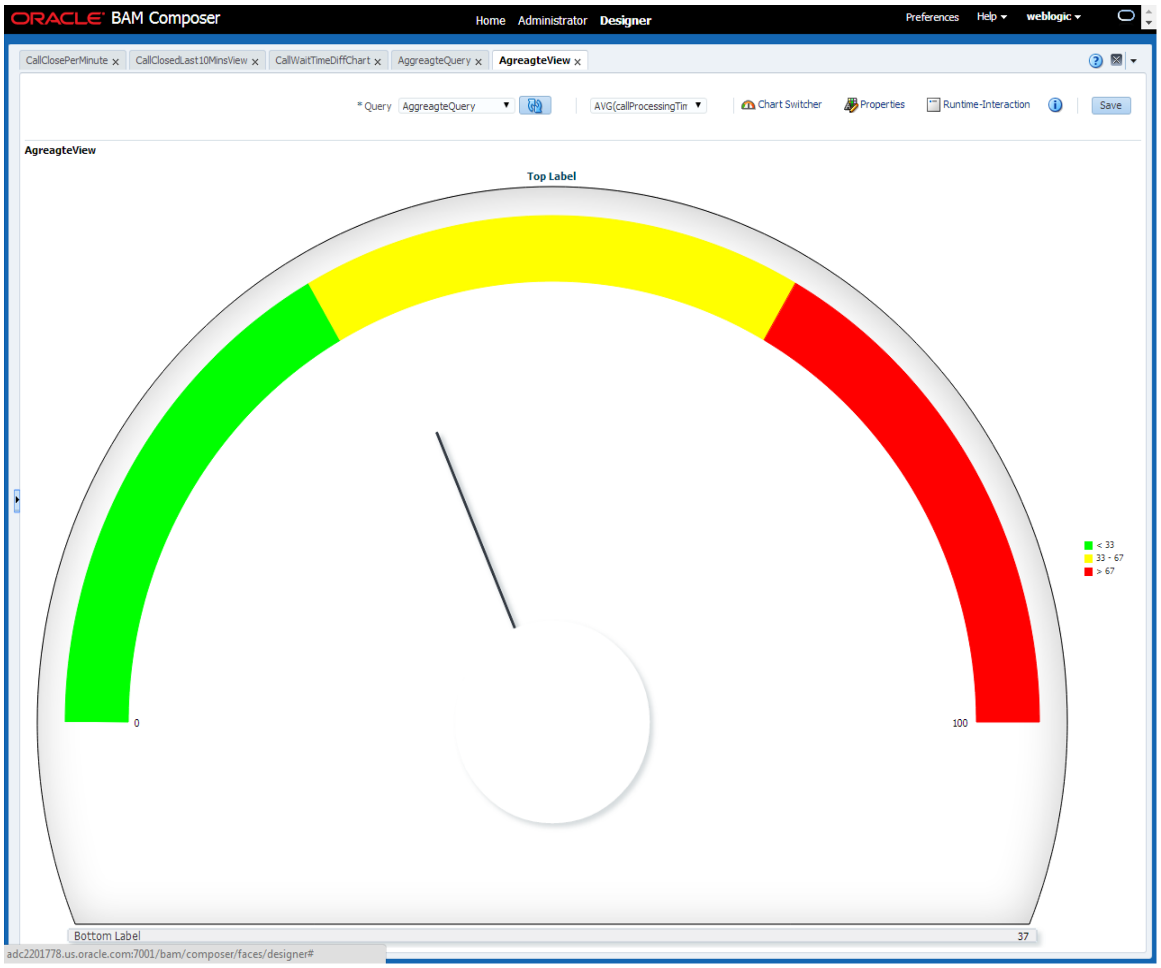


Figure 4 – Gauge meter view

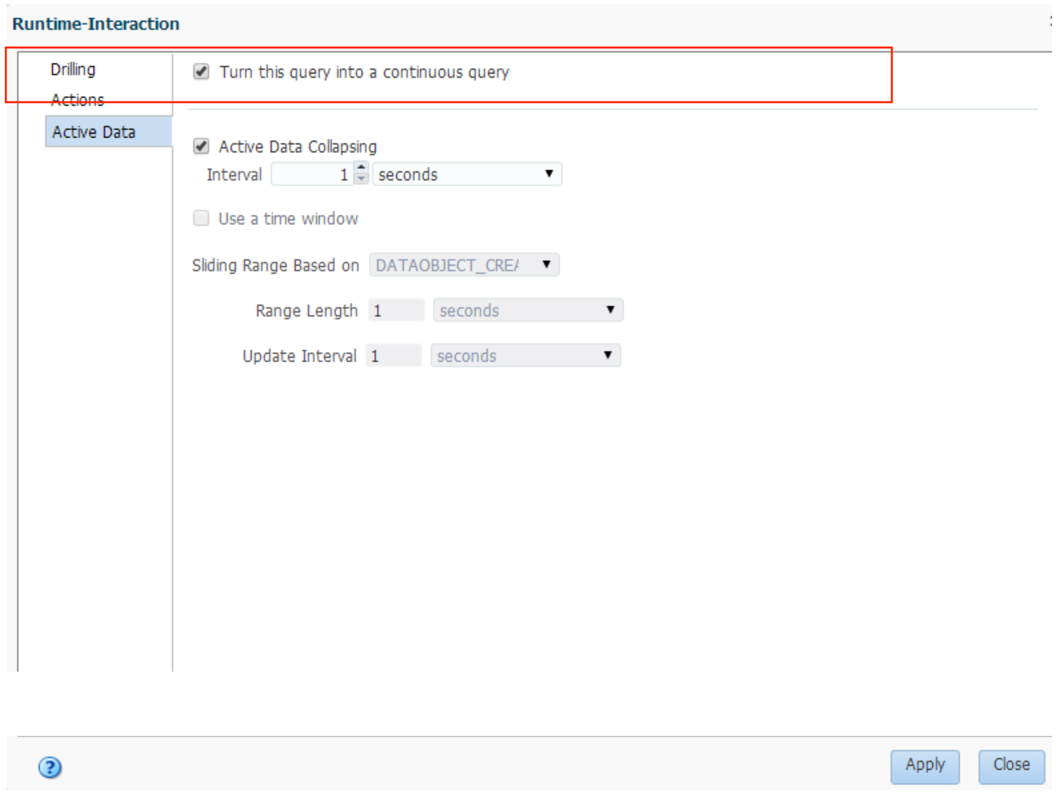
## Understanding Business Views

Business views can be tactical or active. Any static (tactical) dashboard like a group query or a flat query can be converted to an active dashboard using the following steps:

1. Open the gear icon on right-hand corner of the dashboard. This opens a window as shown in Figure 5.
2. Click the “Turn this query into a continuous query” checkbox. To control the active data output, enable the time interval window or the “Active Data Collapsing” checkbox, as shown in Figure 6.

   You can throttle the active data frequency using the Active Data Collapsing feature. The dashboard will show active data only at the specified interval, if any. Active Data Collapsing also ensures that data activity on business views is configured to show significant live data pertinent to your needs. You can filter data by clicking the ‘Runtime Filters and Active Data Properties’ gear icon at the top right corner of the dashboard.

You can enable the active mode in two situations: At design time (edit mode) while creating or editing a business view, or when a dashboard is opened. If you enable it during design mode, all the dashboards embedded in this business view are affected. If enabled on an opened dashboard, only the specific dashboard is ADS enabled.


Runtime-Interaction

- Drilling
- Actions
- Active Data

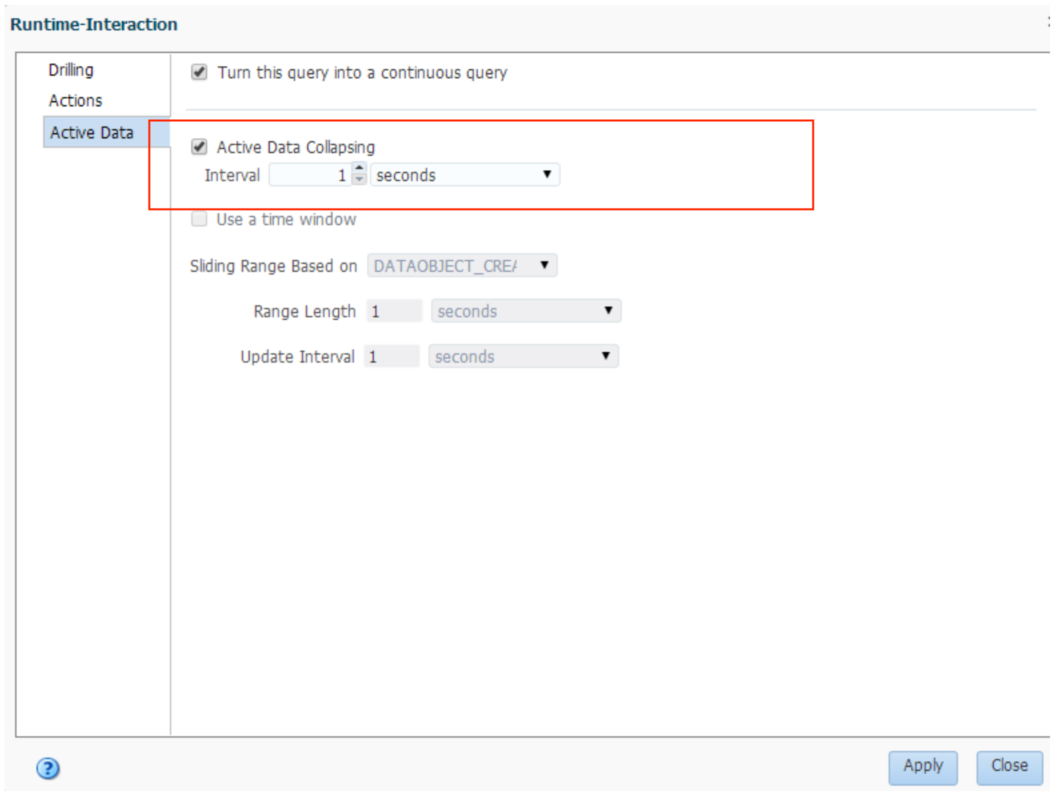
[✓] Turn this query into a continuous query

[✓] Active Data Collapsing
Interval 1 seconds

[ ] Use a time window

Sliding Range Based on DATAOBJECT_CRE/
Range Length 1 seconds
Update Interval 1 seconds

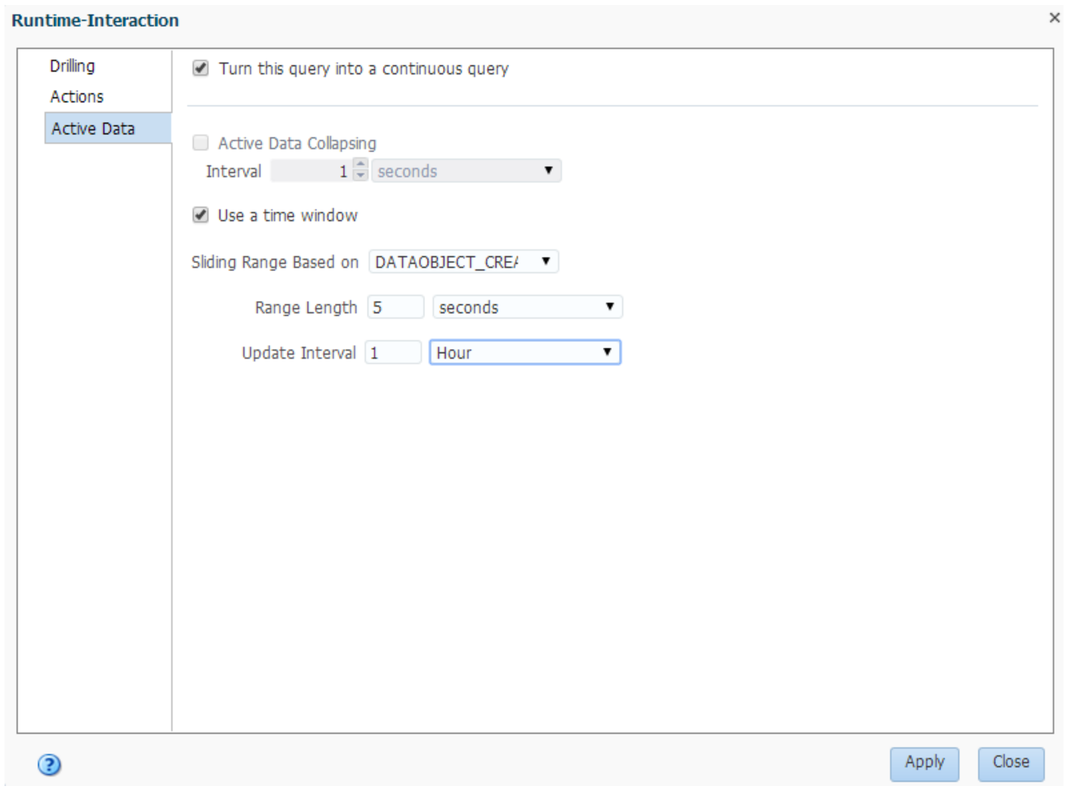
Figure 5 – Turning a static query into an active query


Runtime-Interaction

- [✓] Turn this query into a continuous query
- [✓] Active Data Collapsing
  - Interval: 1 seconds
- [ ] Use a time window
  - Sliding Range Based on: DATAOBJECT_CRE/
  - Range Length: 1 seconds
  - Update Interval: 1 seconds

Figure 6 – Active Data Collapsing

On choosing Active Data Collapsing (i.e without a time window), data is collapsed or the aggregate function is applied at every interval specified. The recommended value for Active Data Collapsing interval is five seconds for bar and pie charts and ten seconds or more for gauges as gauge meter views require refreshes.


Runtime-Interaction

- [✓] Turn this query into a continuous query
- [ ] Active Data Collapsing
- Interval: 1 seconds
- [✓] Use a time window
- Sliding Range Based on: DATAOBJECT_CRE/
- Range Length: 5 seconds
- Update Interval: 1 Hour

Figure 7 – Time Window

With a time window, we measure a moving window, e.g., avg(FilmSales) over the last 60 minutes) and the update interval specifies the output interval (throttle).

## Running a FilmSales Dashboard

To fully understand Oracle BAM dashboards, we introduce an example Film Sales company scenario. Imagine that you are now running a FilmSales Dashboard using the following steps.

1. Import the film sales project using the following bamcommand:

   ./bamcommand -username weblogic -host localhost -port 7003 -cmd import –file FilmSales.zip -type project -name "FilmSales".

2. Ensure that the BAM project file FilmSales.zip is available. Once you import the project, open the FilmSalesDB Dashboard to see a similar dashboard as shown in Figure 6. Open the dashboard to simulate the data by pushing some sample data into the FilmSales DO through loadgen:

   ./bamloadgen -XMLFile FilmSalesLoadGen.xml -frequency 1 -batch_size 1 -duration 20 -host <yourhost> -port <your port>

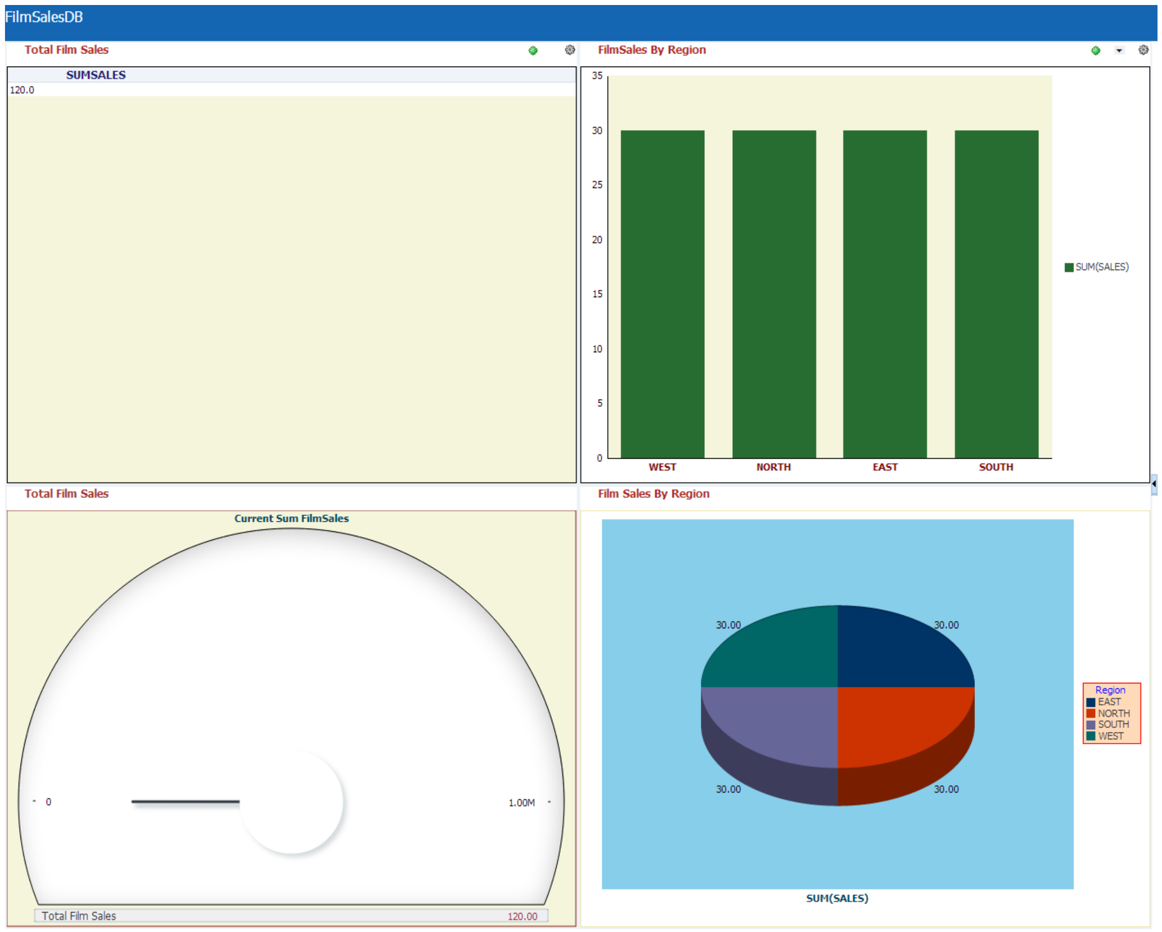


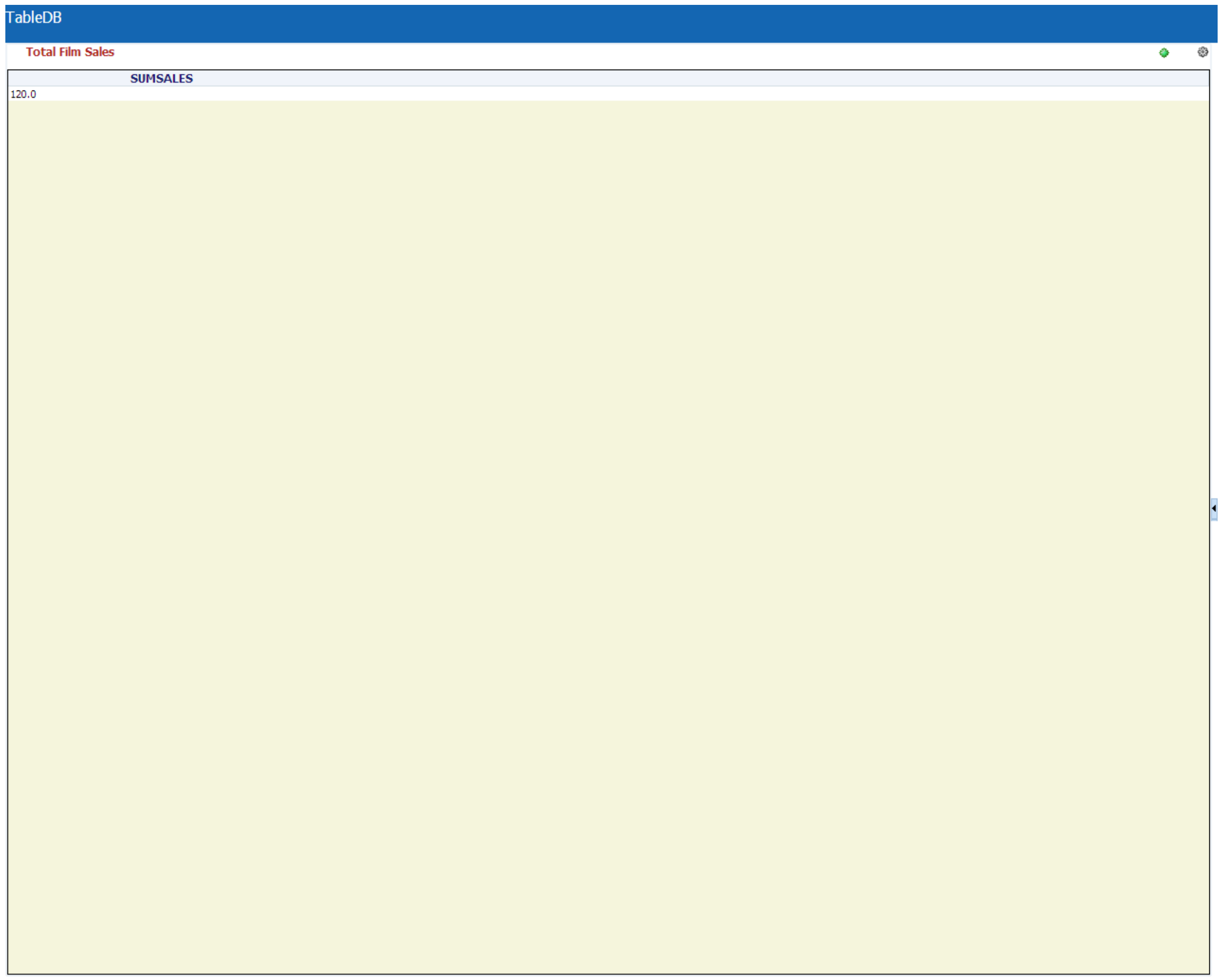
Figure 8 – Film Sales Dashboard and associated business views

There are four visualizations depicted in Figure 8 that allow you to read and interpret data. They are as follows. The gauge meter view and the pie chart view are described in:

1. **TableDB**: This table shows overall sales across all regions. It updates in real time as and when new sales data starts to come in to the system. The CQL query that is registered is:

   CREATE QUERY TotalSale_V1_0_SN18 as SELECT SUM(SALES) AS SUMSALES , 1 AS _internal_id FROM FilmSales EVALUATE EVERY 1 SECONDS PRIMARY KEY ( _internal_id) destination "jms:queue/oracle.beam.cqservice.mdbs.reportcache:queuecf/oracle.beam.cqservice.mdbs.reportcache?batch=true,semantic=update".

   The total sales data is calculated across all the records in a data object, as shown in Figure 9.


TableDB

Total Film Sales

| SUMSALES |
|---|
| 120.0 |

Figure 9 – Table DB showing total film sales.

2. **FilmSales by Region:** You can visualize the entire sum of sales data grouped by region across NORTH, SOUTH, EAST, AND WEST in this case. The query created is:

   “CREATE QUERY FilmSalesQuery_V1_0_SN100 as SELECT REGION AS REGION, SUM(SALES) AS SUMSALES FROM FilmSales GROUP BY REGION EVALUATE EVERY 1 SECONDS PRIMARY KEY ( REGION) destination "jms:queue/oracle.beam.cqservice.mdbs.reportcache:queuecf/oracle.beam.cqservice.mdbs.reportcache?batch=true,semantic=update".

   Here, the EVALUATE clause limits the active data interval to one second and GROUP BY REGION groups the result into the available regions in the data, as shown in Figure 10.

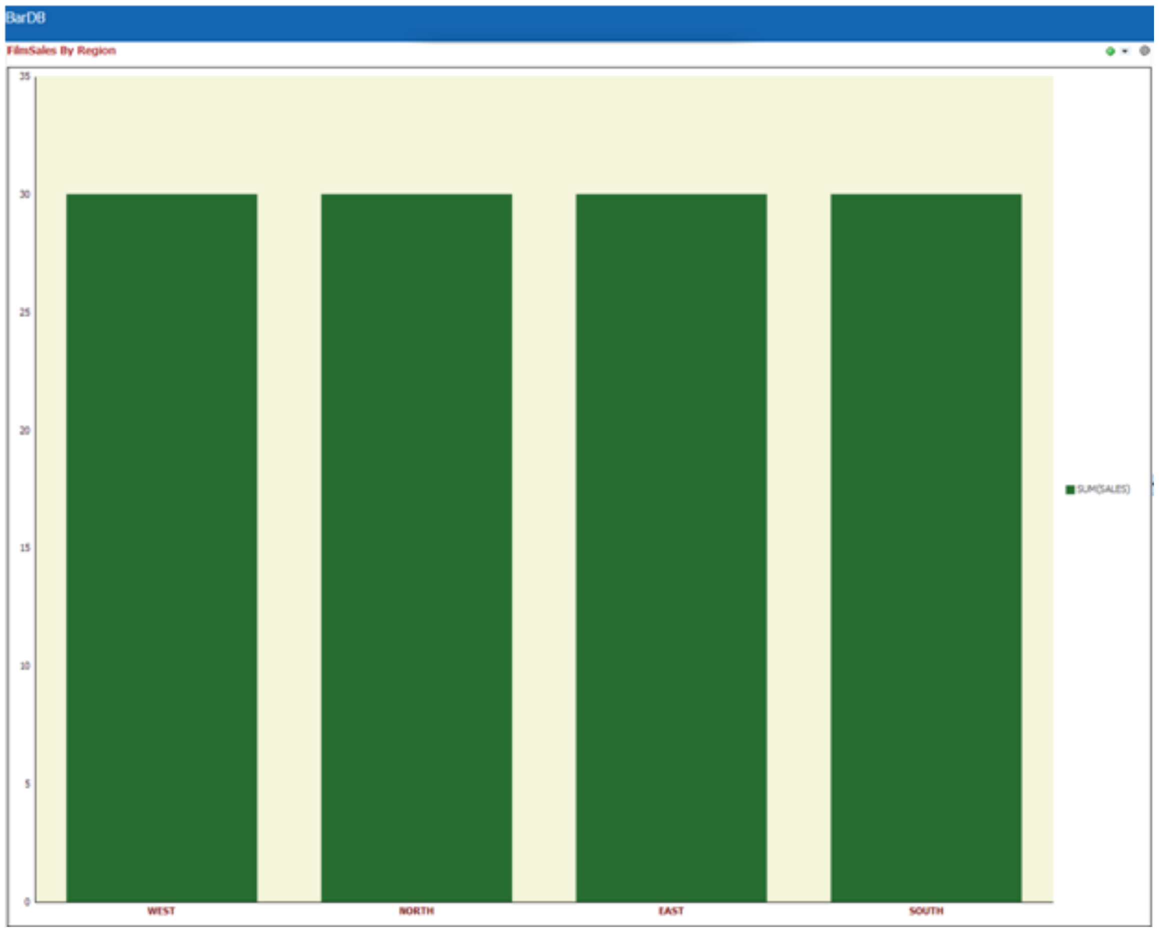
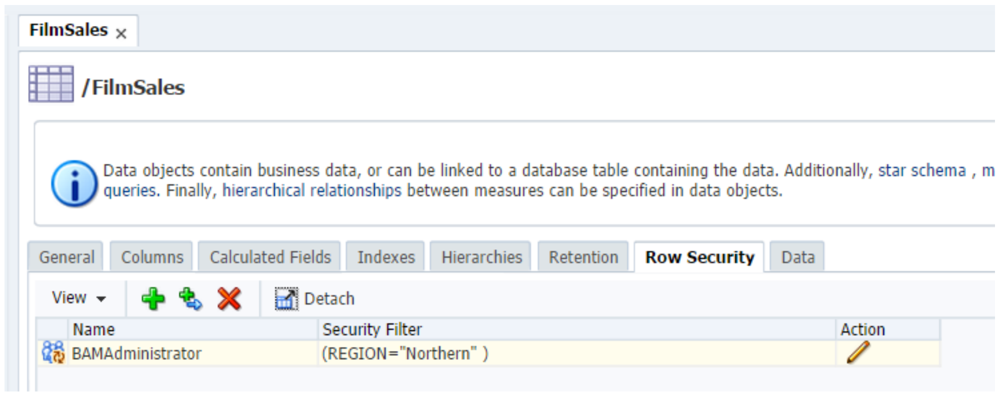


Figure 10 – Film Sales by Region

You can also apply the row security filter as shown in Figure 11. This is to restrict access to data based on user privileges.

In order to create row security filter, go to the Administrator -> FilmSales -> RowSecurity

In this use case, a row security filter for BAMAdministrator where REGION column equals “Northern” is created. This means that all the users in this role can only see those rows where the REGION column has a value of “Northern”.


FilmSales ×

/FilmSales

Data objects contain business data, or can be linked to a database table containing the data. Additionally, star schema , m
queries. Finally, hierarchical relationships between measures can be specified in data objects.

| Name | Security Filter | Action |
|---|---|---|
| BAMAdministrator | (REGION="Northern" ) | |

Figure 11 – Applying the Row-Level Security Filter

As seen in the below Figure 12, we can see a bar only for Northern value

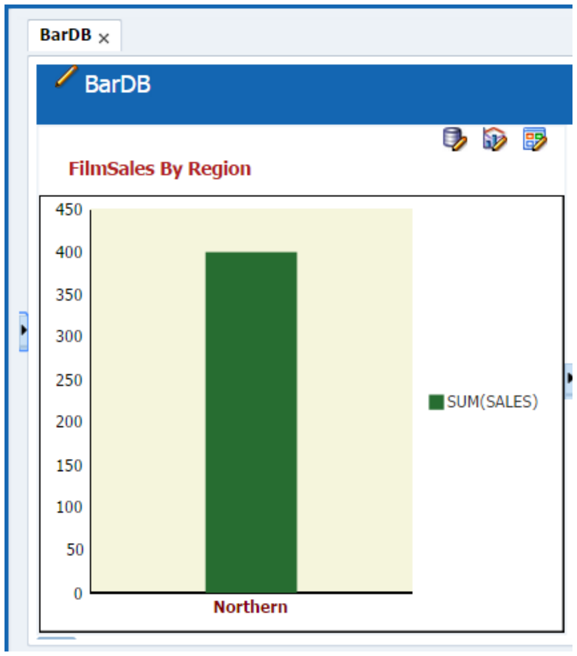
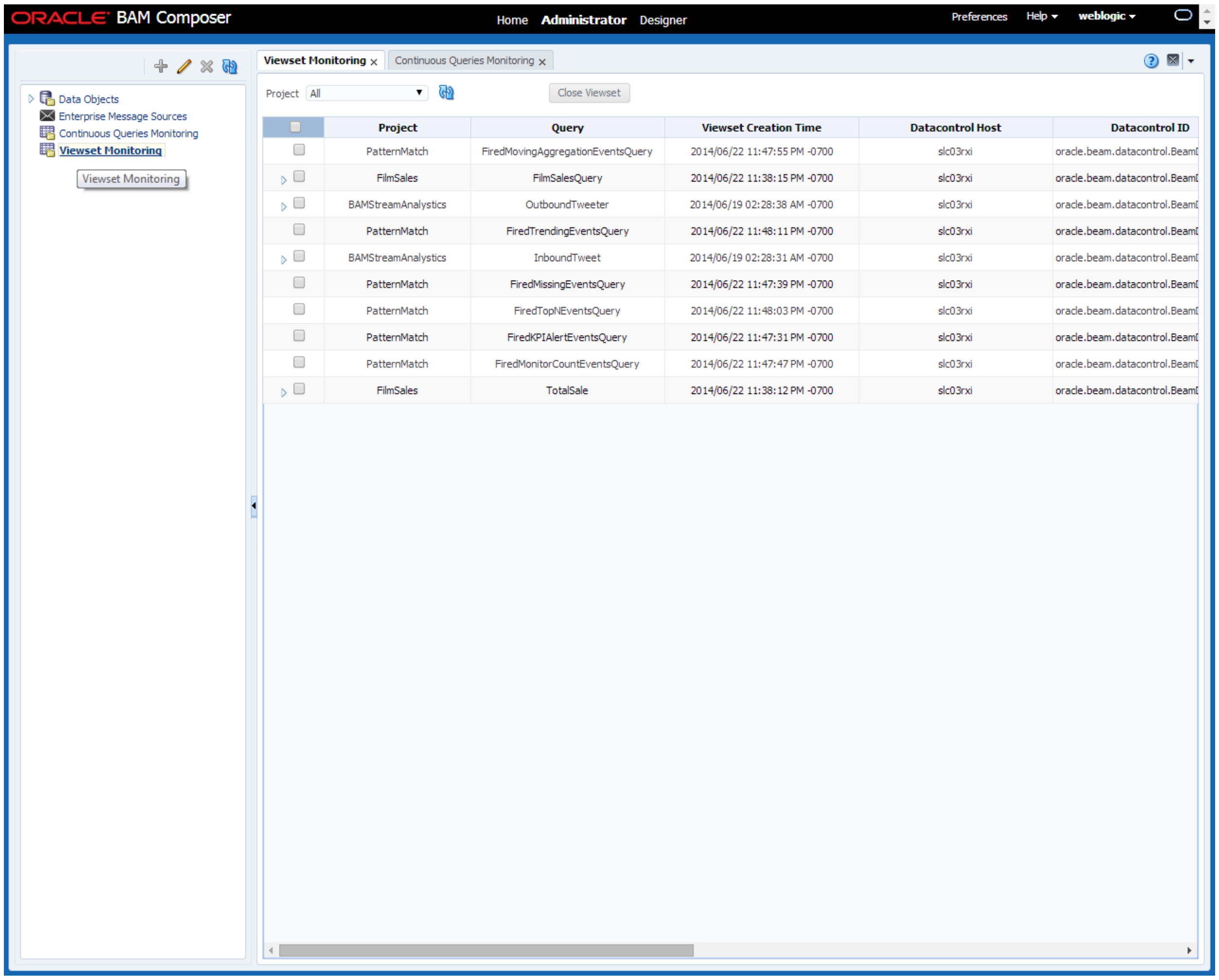


Figure 12 – Film Sales by Region with Row-Level Security Applied

Applying row security filters creates multiple view sets. If two users belong to two different roles and row security filters are applied for each of these roles, this will create two different view sets if both the users are looking at the same view.

**Monitoring Active viewsets:** Once a dashboard is opened, the corresponding active viewsets are opened in BAM (in the ReportCache module). These viewsets can be monitored for their various characteristics isolating the master viewset and counting the slave viewsets that are opened for the master viewset in the BAM Admin section as shown in Figure 13. You can single out the BAM Project that the viewset belongs to, the business query name, the continuous query name (CQL Query), viewset ID, last pinged time by DataControl to keep the viewset from expiring, and the DataControl host that has opened this viewset. Each of these viewsets represents the master viewset and can have many slave viewsets that are visible when you expand the master viewset.

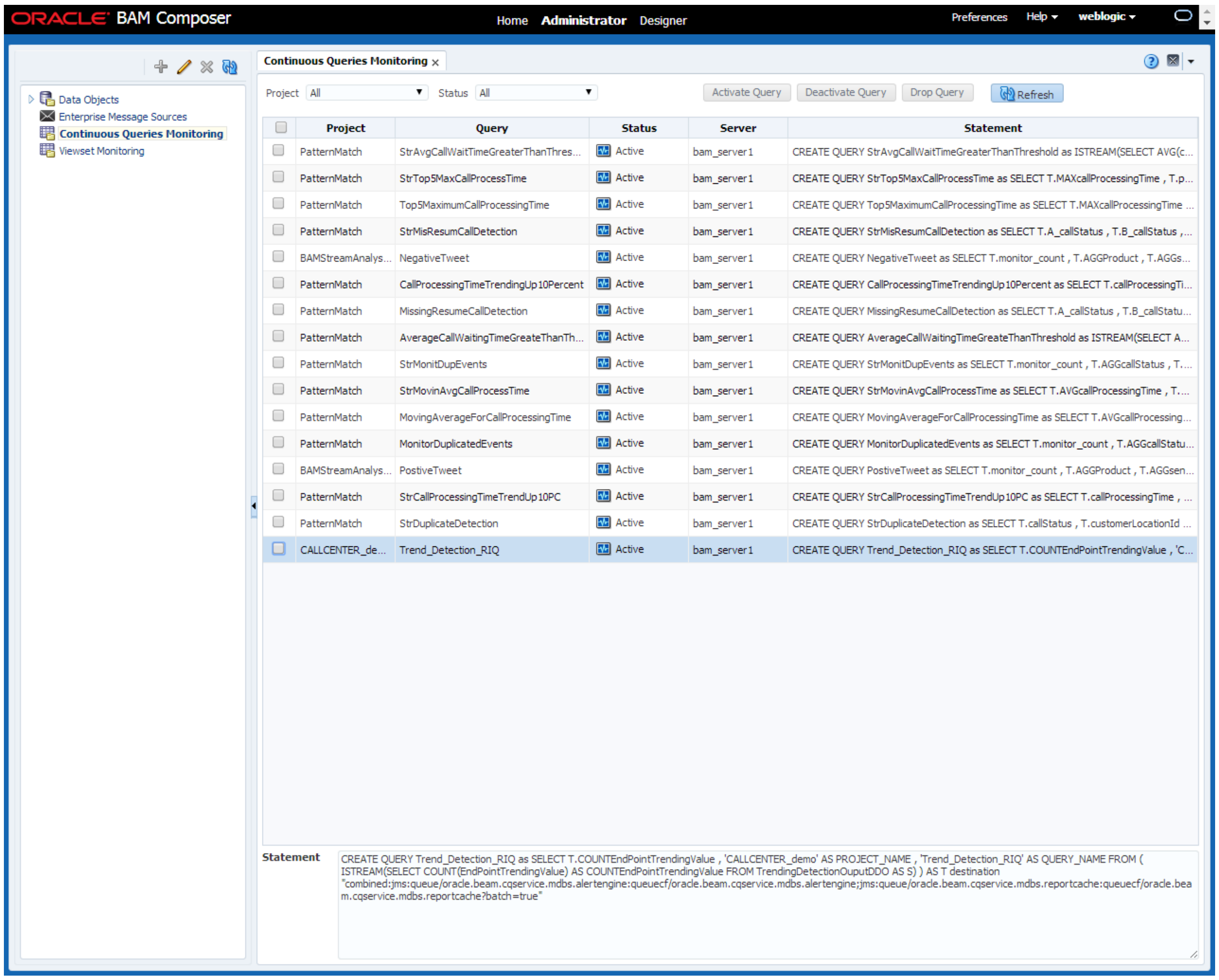
Viewsets get deleted on closing the view. You can also manually select the view sets and click on close viewset.



| | Project | Query | Viewset Creation Time | Datacontrol Host | Datacontrol ID |
|---|---|---|---|---|---|
| | PatternMatch | FiredMovingAggregationEventsQuery | 2014/06/22 11:47:55 PM -0700 | slc03rxi | oracle.beam.datacontrol.Beam[ |
| | FilmSales | FilmSalesQuery | 2014/06/22 11:38:15 PM -0700 | slc03rxi | oracle.beam.datacontrol.Beam[ |
| | BAMStreamAnalystics | OutboundTweeter | 2014/06/19 02:28:38 AM -0700 | slc03rxi | oracle.beam.datacontrol.Beam[ |
| | PatternMatch | FiredTrendingEventsQuery | 2014/06/22 11:48:11 PM -0700 | slc03rxi | oracle.beam.datacontrol.Beam[ |
| | BAMStreamAnalystics | InboundTweet | 2014/06/19 02:28:31 AM -0700 | slc03rxi | oracle.beam.datacontrol.Beam[ |
| | PatternMatch | FiredMissingEventsQuery | 2014/06/22 11:47:39 PM -0700 | slc03rxi | oracle.beam.datacontrol.Beam[ |
| | PatternMatch | FiredTopNEventsQuery | 2014/06/22 11:48:03 PM -0700 | slc03rxi | oracle.beam.datacontrol.Beam[ |
| | PatternMatch | FiredKPIAlertEventsQuery | 2014/06/22 11:47:31 PM -0700 | slc03rxi | oracle.beam.datacontrol.Beam[ |
| | PatternMatch | FiredMonitorCountEventsQuery | 2014/06/22 11:47:47 PM -0700 | slc03rxi | oracle.beam.datacontrol.Beam[ |
| | FilmSales | TotalSale | 2014/06/22 11:38:12 PM -0700 | slc03rxi | oracle.beam.datacontrol.Beam[ |

Figure 13 – Oracle BAM Composer Viewset Monitoring

**Monitoring Continuous Queries:** Once you have opened an active dashboard, apart from monitoring the viewsets, you can monitor continuous queries created by these active viewsets from the "Continuous Query Monitoring" link on the Administrator page, as shown in Figure 14. It shows all the continuous queries that are registered in the BAM system and their statuses. You can also see the pertinent project, query name, server, and the query statement.



Project All Status All

| | Project | Query | Status | Server | Statement |
|---|---|---|---|---|---|
| ☐ | PatternMatch | StrAvgCallWaitTimeGreaterThanThres... | Active | bam_server1 | CREATE QUERY StrAvgCallWaitTimeGreaterThanThreshold as ISTREAM(SELECT AVG(c... |
| ☐ | PatternMatch | StrTop5MaxCallProcessTime | Active | bam_server1 | CREATE QUERY StrTop5MaxCallProcessTime as SELECT T.MAXcallProcessingTime , T.p... |
| ☐ | PatternMatch | Top5MaximumCallProcessingTime | Active | bam_server1 | CREATE QUERY Top5MaximumCallProcessingTime as SELECT T.MAXcallProcessingTime ... |
| ☐ | PatternMatch | StrMisResumCallDetection | Active | bam_server1 | CREATE QUERY StrMisResumCallDetection as SELECT T.A_callStatus , T.B_callStatus ,... |
| ☐ | BAMStreamAnalys... | NegativeTweet | Active | bam_server1 | CREATE QUERY NegativeTweet as SELECT T.monitor_count , T.AGGProduct , T.AGGs... |
| ☐ | PatternMatch | CallProcessingTimeTrendingUp10Percent | Active | bam_server1 | CREATE QUERY CallProcessingTimeTrendingUp10Percent as SELECT T.callProcessingTi... |
| ☐ | PatternMatch | MissingResumeCallDetection | Active | bam_server1 | CREATE QUERY MissingResumeCallDetection as SELECT T.A_callStatus , T.B_callStatu... |
| ☐ | PatternMatch | AverageCallWaitingTimeGreateThanTh... | Active | bam_server1 | CREATE QUERY AverageCallWaitingTimeGreateThanThreshold as ISTREAM(SELECT A... |
| ☐ | PatternMatch | StrMonitDupEvents | Active | bam_server1 | CREATE QUERY StrMonitDupEvents as SELECT T.monitor_count , T.AGGcallStatus , T.... |
| ☐ | PatternMatch | StrMovinAvgCallProcessTime | Active | bam_server1 | CREATE QUERY StrMovinAvgCallProcessTime as SELECT T.AVGcallProcessingTime , T.... |
| ☐ | PatternMatch | MovingAverageForCallProcessingTime | Active | bam_server1 | CREATE QUERY MovingAverageForCallProcessingTime as SELECT T.AVGcallProcessing... |
| ☐ | PatternMatch | MonitorDuplicatedEvents | Active | bam_server1 | CREATE QUERY MonitorDuplicatedEvents as SELECT T.monitor_count , T.AGGcallStatu... |
| ☐ | BAMStreamAnalys... | PostiveTweet | Active | bam_server1 | CREATE QUERY PostiveTweet as SELECT T.monitor_count , T.AGGProduct , T.AGGsen... |
| ☐ | PatternMatch | StrCallProcessingTimeTrendUp10PC | Active | bam_server1 | CREATE QUERY StrCallProcessingTimeTrendUp10PC as SELECT T.callProcessingTime , ... |
| ☐ | PatternMatch | StrDuplicateDetection | Active | bam_server1 | CREATE QUERY StrDuplicateDetection as SELECT T.callStatus , T.customerLocationId ... |
| ☐ | CALLCENTER_de... | Trend_Detection_RIQ | Active | bam_server1 | CREATE QUERY Trend_Detection_RIQ as SELECT T.COUNTEndPointTrendingValue , 'C... |

**Statement**

```
CREATE QUERY Trend_Detection_RIQ as SELECT T.COUNTEndPointTrendingValue , 'CALLCENTER_demo' AS PROJECT_NAME , 'Trend_Detection_RIQ' AS QUERY_NAME FROM (
ISTREAM(SELECT COUNT(EndPointTrendingValue) AS COUNTEndPointTrendingValue FROM TrendingDetectionOuputDDO AS S) ) AS T destination
"combined:jms:queue/oracle.beam.cqservice.mdbs.alertengine:queuecf/oracle.beam.cqservice.mdbs.alertengine;jms:queue/oracle.beam.cqservice.mdbs.reportcache:queuecf/oracle.bea
m.cqservice.mdbs.reportcache?batch=true"
```

Figure 14 – BAM Composer Continuous Query Monitoring

## Troubleshooting ADS

Oracle BAM has an ADS debugging tool – BAM Diagnostics. Using this tool, you can conduct a component-wise diagnosis. To troubleshoot ADS, enable diagnostics on debug mode for the following components: CQS (CQService), RC (Report Cache), DC (Data Control), and ADF (Active Data Framework). Once enabled, run through the issue scenario to get the appropriate diagnostics data collected. The diagnostics reports are collected in a system level BASE_METRICS DataObject. More specific diagnostics are collected in CQS_METRICS/RC_METRICS DO. In BASE_METRICS, you can visualize API calls and data flowing through ADF-DC-RC-CQS.

### Debugging RC Through Diagnostics Framework

Set the following Mbean for Diagnostics.

1. diagnosticEnabled

   The values supported for this property are “true” and “false”. The default value is “false”.

2. diagnosticLevel

The values supported for this property are “INFO” and “DEBUG”. The default value is “INFO”.

3. RC diagnosticComponents

Make sure you add RC in the list of components ADF, DC, RC, CQS,ALERTS_ENGINE, COMPOSER, COMMON_QUERY.

Each row in the diagnostics starts with the column VIEWSET_ID. This identifies the viewset this row details. The ACTION_NAME column has a series of values indicating the action either DC has taken on RC on that the RC has initiated. These are the following actions in an active viewset lifecycle.

getMessageSelector

openViewSet

Viewset_created

RegisterCQL

SnapshotSent

ActiveDataSent

getRecordSet

Viewset_reset

closeViewSet

Each of these actions has additional details as follows.

PROJECT_NAME – Describes the project which this belongs to.

QUERY_NAME – Describes the business query name.

SENDER_COMPONENT – Indicates which component initiated the call.

RECIEVER_COMPONENT – Indicates which component received the call.

PAYLOAD – Describes additional details like data model payload, query string.

STATUS – ACTIVE/PASSIVE viewset.

EXECUTION_TIME – Indicates total time taken to complete the call.

INSERT_TMSTMP – Indicates time the action happened.

SERVER_NAME – Indicates the server that executed the action.

THREAD_ID – Indicates the thread that executed the action.

SEQUENCE_ID – Shows the last sequence number of the changelist/snapshot processed.

If any of the above actions is missing or if the payload is not as expected through diagnostics data, you can take appropriate measures to resolve the issue as described in the subsequent paragraphs.

Continuous Query Payload : CREATE QUERY collapsedlist.sansgroupBy_V1_0_SN54 as SELECT SUM(Cost) AS SUMCost , 1 AS _internal_id FROM collapsedlist.GourmetFood EVALUATE EVERY 1 SECONDS PRIMARY KEY ( _internal_id) destination "jms:queue/oracle.beam.cqservice.mdbs.reportcache:queuecf/oracle.beam.cqservice.mdbs.report cache?batch=true,semantic=update“

Verify if the query is created as expected.

Ensure that the JMS queue is properly setup in the WLS environment.

You can check the status of the continuous query with the following states:

Registering, Deactivated, Activating, Active, Deactivating, Dropping, Pre-migrating, Migrating.

Snapshot/Changelist payload:

[{SUMCost=606.5, DUMMY_GROUP=0}] Last Sequence Number: 2

Make sure the aggregate value matches the expectation and that group values are as expected.

**DataControl Payload:**

Make sure that the DataControl payload matches sent from DC to ADF matches the one sent from RC to DC.

Data control raising event: oracle.beam.datacontrol.BeamDataControl@7b49747a

#### DataChangeEvent #### on [DataControl name=collapsedlist_sansgroupBy, binding=data.oracle_integration_console_beam_views_ProxyPagePageDef.dynamicRegion1.beam_project_collapsedlist_dashboard_collpasedListDB_taskflow_collpasedListDB_beamDashboardcollpasedListDBPageDef_beam_project_collapsedlist_dashboard_collpasedListDB_taskflow_collpasedListDB_beamDashboardcollpasedListDB_xml_ps_taskflowid.dynamicRegion58.beam_project_collapsedlist_view_collapsedList_taskflow_collapsedList_beamVizcollapsedListPageDef_beam_project_collapsedlist_view_collapsedList_taskflow_collapsedList_beamVizcollapsedList_xml_ps_taskflowid.QueryIterator]

Filter/Collection Id: 0

Collection Level: 0

Sequence Id: 6

ViewSetId: collapsedlist.sansgroupBy_V1_0_SN54

==== DataChangeEntry (#1)

ChangeType: UPDATE

KeyPath: [0, 0]

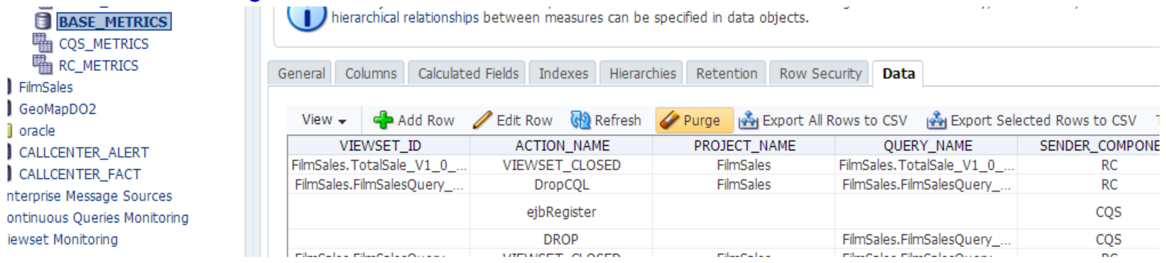
AttributeNames: [SUMCost, DUMMY_GROUP]

AttributeValues: [607.5, 0]

AttributeTypes: [java.lang.Double, java.lang.Integer,  ]

**Using the diagnostic feature**

a) Remove all entries from Base Diagnostics DO by clicking on Purge and select Purge All Rows as shown in Figure 15.



BASE_METRICS
CQS_METRICS
RC_METRICS
FilmSales
GeoMapDO2
oracle
CALLCENTER_ALERT
CALLCENTER_FACT
nterprise Message Sources
ontinuous Queries Monitoring
iewset Monitoring

hierarchical relationships between measures can be specified in data objects.

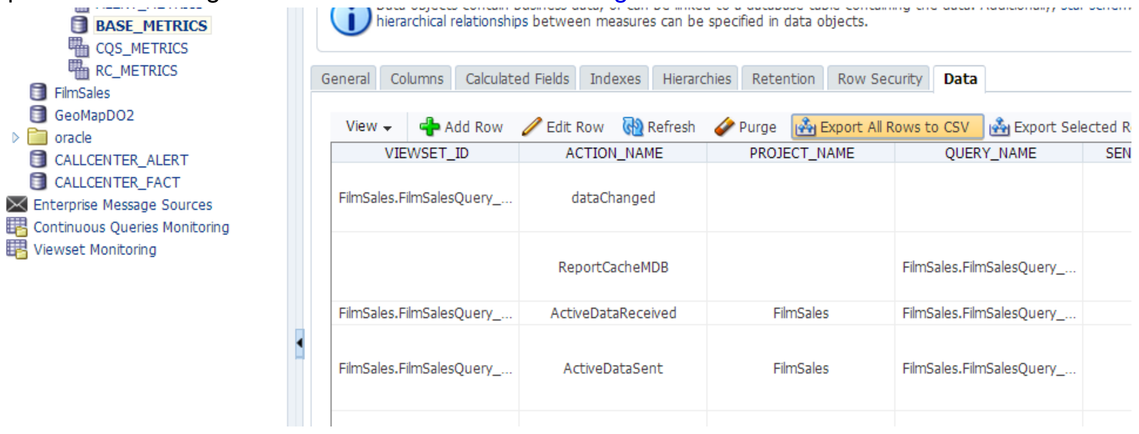
| VIEWSET_ID | ACTION_NAME | PROJECT_NAME | QUERY_NAME | SENDER_COMPONE |
|---|---|---|---|---|
| FilmSales.TotalSale_V1_0_... | VIEWSET_CLOSED | FilmSales | FilmSales.TotalSale_V1_0_... | RC |
| FilmSales.FilmSalesQuery_... | DropCQL | FilmSales | FilmSales.FilmSalesQuery_... | RC |
|  | ejbRegister |  |  | CQS |
|  | DROP |  | FilmSales.FilmSalesQuery_... | CQS |

Figure 15 – Base Metrics Data

b) Turn on Diagnostics as mentioned above.

c) Open the dashboard and run loadgen utility.

d) Export the BaseDiagnosticsDO to CSV as shown in Figure 16.



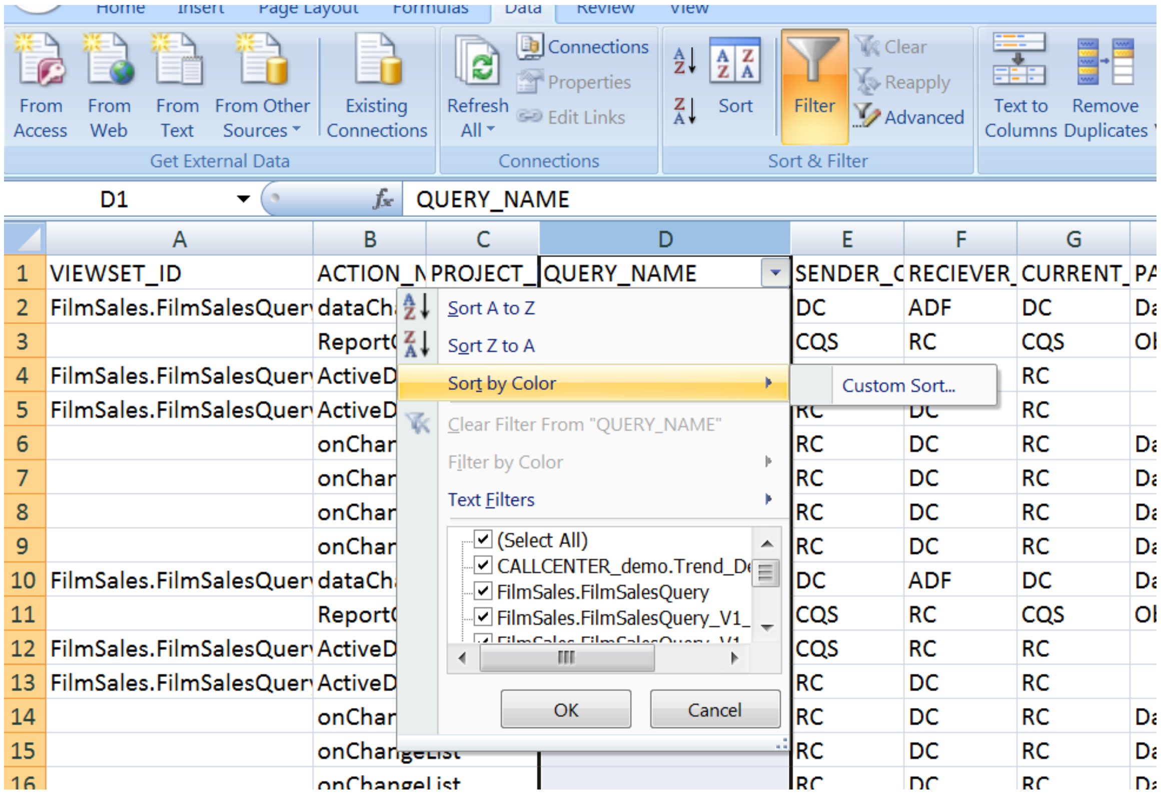
BASE_METRICS
CQS_METRICS
RC_METRICS
FilmSales
GeoMapDO2
oracle
CALLCENTER_ALERT
CALLCENTER_FACT
Enterprise Message Sources
Continuous Queries Monitoring
Viewset Monitoring

hierarchical relationships between measures can be specified in data objects.

| VIEWSET_ID | ACTION_NAME | PROJECT_NAME | QUERY_NAME | SEN |
|---|---|---|---|---|
| FilmSales.FilmSalesQuery_... | dataChanged |  |  |  |
|  | ReportCacheMDB |  | FilmSales.FilmSalesQuery_... |  |
| FilmSales.FilmSalesQuery_... | ActiveDataReceived | FilmSales | FilmSales.FilmSalesQuery_... |  |
| FilmSales.FilmSalesQuery_... | ActiveDataSent | FilmSales | FilmSales.FilmSalesQuery_... |  |

Figure 16 – Exporting Base Metrics Data

e) Open the saved excel file and use filters on columns to look for a particular project or query name.



| | A | B | C | D | E | F | G | |
|---|---|---|---|---|---|---|---|---|
| 1 | VIEWSET_ID | ACTION_N | PROJECT_ | QUERY_NAME | SENDER_C | RECIEVER_ | CURRENT_ | PA |
| 2 | FilmSales.FilmSalesQuer | dataCh | | | DC | ADF | DC | Da |
| 3 | | Report( | | | CQS | RC | CQS | Ol |
| 4 | FilmSales.FilmSalesQuer | ActiveD | | | | | RC | |
| 5 | FilmSales.FilmSalesQuer | ActiveD | | | RC | DC | RC | |
| 6 | | onChar | | | RC | DC | RC | Da |
| 7 | | onChar | | | RC | DC | RC | Da |
| 8 | | onChar | | | RC | DC | RC | Da |
| 9 | | onChar | | | RC | DC | RC | Da |
| 10 | FilmSales.FilmSalesQuer | dataCh | | | DC | ADF | DC | Da |
| 11 | | Report( | | | CQS | RC | CQS | Ol |
| 12 | FilmSales.FilmSalesQuer | ActiveD | | | CQS | RC | RC | |
| 13 | FilmSales.FilmSalesQuer | ActiveD | | | RC | DC | RC | |
| 14 | | onChar | | | RC | DC | RC | Da |
| 15 | | onChangeList | | | RC | DC | RC | Da |
| 16 | | onChangeList | | | RC | DC | RC | Da |

Figure 17 – Example Lookup Using Filters on Base Metric Data