Volume 20 | Number 1
First Quarter 2013

www.ioug.org

SELECT JOURNAL

For the Complete Technology & Database Professional

Manageability

Enterprise Manager 12*c* Cloud Control: What's Changed, What's New

by Michael Messina

Introduction to Oracle Enterprise Manager Command Line Interface

by Ray Smith

Retrieving Large Volumes of Data

by Andrei Dzianisau

IOUG independent oracle users group
20th ANNIVERSARY

If You've Got Your Head in the Clouds You'll Want Your Oracle Database on Cisco UCS

If you are looking for sky-high performance from your Oracle database, coupled with the convenience and cost-effectiveness of a cloud-based delivery model, look no further than the Cisco® Unified Computing System™ (UCS®), Powered by Intel Xeon processors.

Cisco has changed the server game with UCS, through innovations that have transformed data center economics, IT capabilities, and business results for customers like you.

With the industry's fastest and most powerful server for virtualization, Cisco helps you move to a whole new level of performance while reducing space requirements, hardware, cabling, and energy costs.

Powered exclusively by Intel® Xeon® processors backed by a continuous stream of world-record benchmarks, UCS is the only non-Oracle/Sun hardware certified with NoSQL by Oracle and the only hardware that Oracle has chosen for two-socket TPC-C performance partnership.

Find out more now at www.cisco.com/go/oracle

DISTINGUISHED IOUG independent oracle users group ALLIANCE MEMBER

How can I contribute to *SELECT Journal*?
Write us a letter. Submit an article. Report on Oracle use in all corners of the globe. We prefer articles that conform to APA guidelines. Send to select@ioug.org.

Headquarters
Independent Oracle Users Group
330 North Wabash Avenue
Chicago, IL 60611-7621
USA
Phone: +1.312.245.1579
Fax: +1.312.527.6785
E-mail: ioug@ioug.org

Editorial
Theresa Wojtalewicz
Managing Editor
IOUG Headquarters
Phone: +1.312.673.5870
Fax: +1.312.245.1094
E-mail: twojtalewicz@ioug.org

How do I get the next one?
SELECT Journal is a benefit to members of the Independent Oracle Users Group. For more information, contact IOUG Headquarters at +1.312.245.1579

***SELECT Journal* Online**
For the latest updates and addendums or to download the articles in this and past issues of *SELECT Journal*, visit www.selectjournal.org.

CONTENTS

Volume 20, No. 1, 1st Qtr. 2013

Features

Reviewers for This Issue

Syamal Bandyopadhyay
Youry Khmelevsky
Tim Covey
Satishbabu Gunkula
Wilquins Charleston
Lee Murray
Dan Hotka
James Voigt

Regular Features

Manageability in the New Year

April Sims
Executive Editor

This edition of the *SELECT Journal* is focused on manageability with several articles focused on the newest edition of Oracle Enterprise Manager 12*c* Cloud Control. Mike Messina gives us a quick look at what has changed in this OEM with his article "Enterprise Manager 12*c* Cloud Control: What's Changed, What's New" as a heads up before you migrate to this new version. Ray Smith brings to light one of most underused utilities for OEM called the Command Line Interface, or OEMCLI, in "Introduction to Oracle Enterprise Command Line Interface." It should be part of your toolkit to optimize the use of OEM across your enterprise, improving the efficiency of repetitive tasks versus the point-and-click of the GUI interface.

Also in this issue, it's back to basics on skills with advice from Andrei Dzianisau in his article "Retrieving Large Volumes of Data." Tom Sager also delves into achieving high-availability without using Real Application Clusters in "A Multilayered Approach to Oracle Database Availability."

It is my job to make sure you, the reader, take something away from each of these articles no matter your role in the organization. I look forward to meeting with readers at IOUG's upcoming annual conference, COLLABORATE 13. Please provide feedback on how we are doing!

This issue also includes a new installation of the Oracle ACE Spotlight featuring Gokhan Atil. In this section, *SELECT Journal* gets the opportunity to highlight certain members of IOUG. Expect to witness a small, personal glimpse into what has been important to the professional careers of those featured.

2013 Themes and Topics

We are actively looking for article, features and discussions related to the following topics for next year. Other themes will be announced as they are finalized.

Q2: Reducing Risk — Is your enterprise able to deploy quarterly security updates in a timely manner?

Q3: Consolidation — With the anticipated release of 12*g* RDBMS as a pluggable containerized database, this issue starts the deep dive into this new release.

Once again, we would like to thank all our authors and reviewers for their efforts in providing high-quality content for *SELECT Journal*. We always welcome your input and feedback, and we look forward to you sharing your expertise via this fine medium. Email us at **select@ioug.org** if you would like to submit an article or sign up to become reviewer.

April Sims
Executive Editor, IOUG SELECT Journal

An Exciting Year Ahead

John Matelski
IOUG President

Happy 20th anniversary, IOUG! 2013 marks two decades of existence for this outstanding user community and also heralds the start of many new and exciting things for the association.

As the voice of the Oracle technology user community, IOUG has been, and will continue to be, directly involved in providing feedback and insight into Oracle. In January, numerous IOUG community members participated in a weeklong beta testing program for the next generation of Oracle database in the interest of helping to influence product direction, documentation and the overall value for the community.

Additionally, IOUG's ResearchWire program continues to influence and inform the Oracle technology community. By soliciting feedback directly from Oracle users who live and experience front-line challenges on a daily basis, IOUG ResearchWire studies influence future business strategy, IT spending and the content of training at events like the upcoming COLLABORATE 13 – IOUG Forum. The most recent study, which focuses on the emerging role of the data scientist, revealed that more and more data professionals are being tapped to "tell a story" with their data and to drive business decision making. Keep an eye out throughout the next month or two for this forthcoming study, as well as related sessions at COLLABORATE 13.

Speaking of COLLABORATE 13, the annual user conference is fast approaching, taking place this April 7-11 in Denver. To stay ahead of the curve on the latest topics, the IOUG Forum has it all. From sessions designed for both new and advanced DBAs; to user experience-driven case studies on manageability and engineered systems; to the Senior IT Leadership Program — this brand-new, IOUG-exclusive curriculum will provide attendees with skills to enable them to influence, present business cases and help develop business solutions for their company. This program kicks off that Sunday with an expert panel covering "The Essence of Great Leadership" and continues through the week with sessions led by an impressive lineup of Oracle pros.

Beyond COLLABORATE, in the year of our 20th anniversary, IOUG delivers an even greater value to its membership. Look for more ways to access leading practices in our year-round virtual education program, an expansion of *SELECT Journal* online (including author videos and letters to the editor), an improved tip booklet, as well as new e-news briefs on enterprise management topics. As always, IOUG encourages and welcomes additional contributors to *SELECT Journal*. Express your interest in becoming an author or reviewer by emailing **select@ioug.org**.

On behalf of the IOUG, I'm really looking forward to another fantastic year for the community.

John Matelski
IOUG President

VISIBILITY IS KEY

VMware plugin for EM12c lets you see it all at once

- Extends Oracle EM12c to VMware
- A single integrated picture of Oracle on VMware – fast and easy
- Visibility – see into each layer of your virtualized Oracle environment

For more information, visit
bluemedora.com/em-products

bluemedora

The leader in extending Oracle Enterprise Manager

Enterprise Manager 12*c* Cloud Control

What's Changed, What's New

By Michael Messina
Edited by Ray Smith

The role and tasks performed by Oracle Enterprise Manager (OEM) evolved with the expansion of Enterprise Manager capabilities, while leveraging OEM for Oracle's growing application portfolio. Shortly after Oracle purchased BEA Systems Inc., a team was assembled to reinvent OEM — from its architecture to its interface. These changes were significant, and it is important to recognize the pivotal role in which OEM capabilities play in Oracle's overall product strategy.

The OEM product began as tool for DBAs and developers. It was a simple console for managing a single database. With each new product release, the administrators' toolkit had to grow into an increasingly scalable and stable product. By the time Oracle Database 10*g* was released, it was an all-encompassing tool for the IT staff.

Product placement moved OEM from the database DBA's tool into a central management console where businesses can monitor and analyze the total user experience from beginning to end.

This article will detail the shifts of OEM architecture and product placement and how we administer and use Oracle Enterprise Manager 12*c* Cloud Control.

What's Changed?

The change in architecture has changed the look and process flows of OEM as illustrated below.

Logging In

When connecting to OEM 12*c*, notice the screen has a new look, resembling the Oracle WebLogic Server console and Fusion Server login screens (Figure 1).

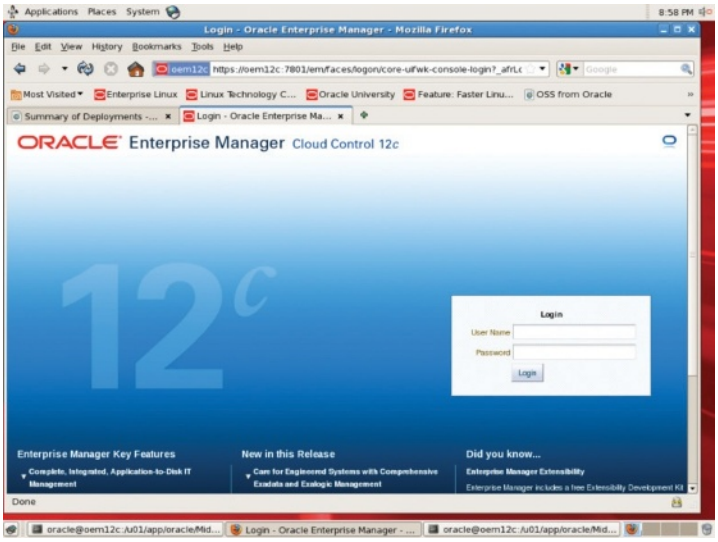


Figure 1: Login Screen

The initial login includes the usual license and management pack authorization page. Read the license information carefully and select "Accept" to continue (Figure 2).

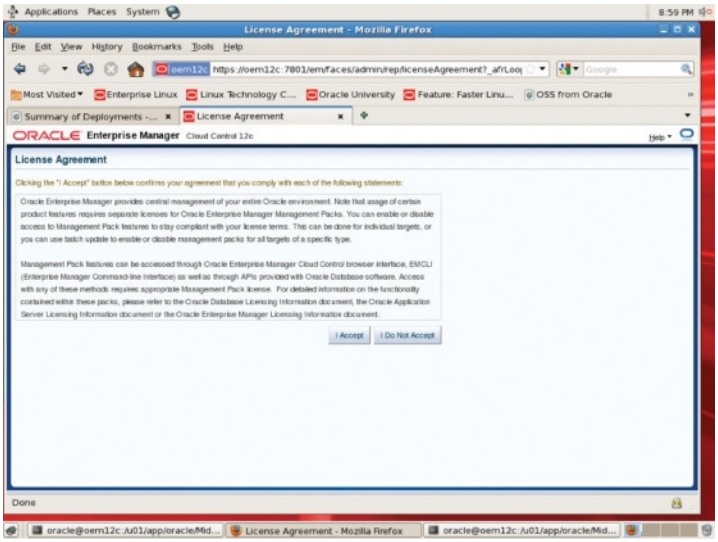


Figure 2: License Information

Interface Changes

There are several interface changes inside the OEM 12*c* console that affect navigation, location of target screens, preferences and the ability to customize screen looks and information. This section will highlight the more noticeable changes and the ones most likely to affect your OEM 12*c* console experience as an end user.

As you explore the home screens, it's important to note that the selection of a home screen has no effect on the user's ability to access or use other functionalities within OEM; the home page is merely a starting point.

continued on page 6

Home Screen Selection

Each OEM user may configure and reconfigure their own home screen based on personal preferences, current activities and job role. During the initial login for each account, the user is prompted to select a default view from six basic designs, which can be changed at any time. (See Appendix A, Changing Home View). The most commonly utilized views for the DBA or developer are described below.

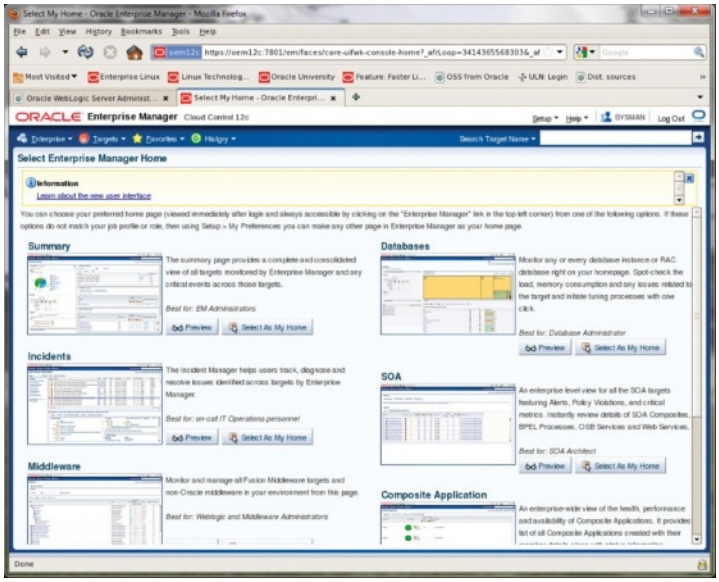


Figure 3: Home Screen Selection

Summary View

The summary view looks the most like the home screen of Oracle Enterprise Manager Grid Control. This view is ideal for enterprise manager level administrators, allowing the administrator visibility of many target types. This is the best view to give the super administrator SYSMAN account, as SYSMAN would be considered an enterprise manager administrator.

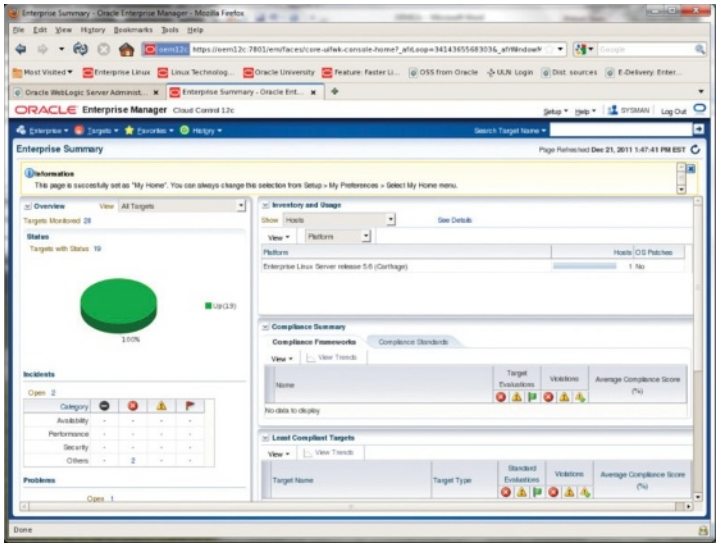


Figure 4: Summary View

Databases View

As the name suggests, the databases view would be most appealing to administrators whose primary responsibilities are centered around databases with little interaction with other target types. There are two layouts available for the databases view: Oracle Load Map (Figure 5) and Search List (Figure 6). Users can switch between the views with a single button click.

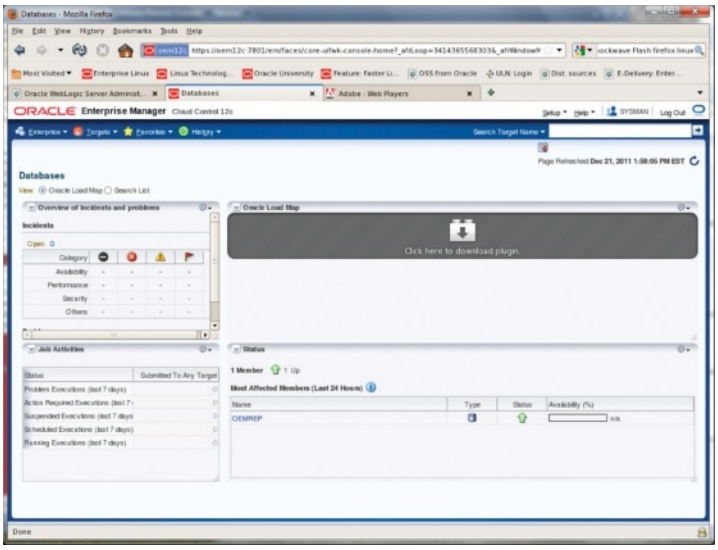


Figure 5: Oracle Load Map

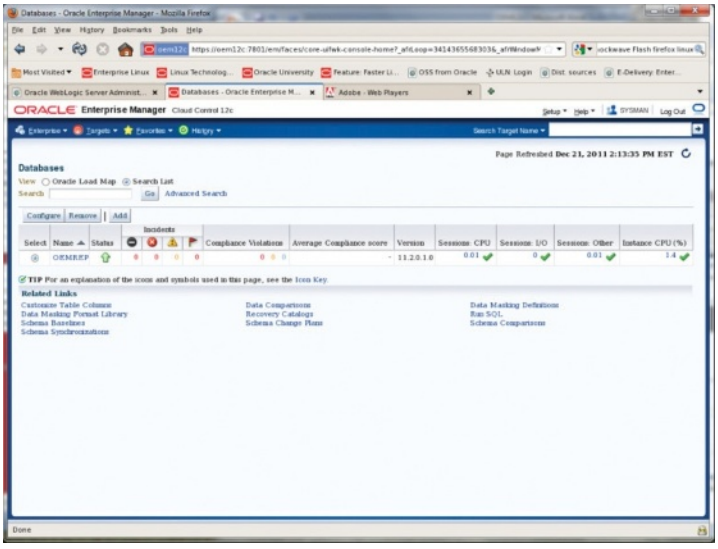


Figure 6: Search List

Incidents View

The incidents view allows quick, easy display of the open incidents in the monitored environment that includes all target types. This view is beneficial for operations, call center operators and console monitors (Figure 7). The improved incident management features of OEM 12*c* will be further detailed.

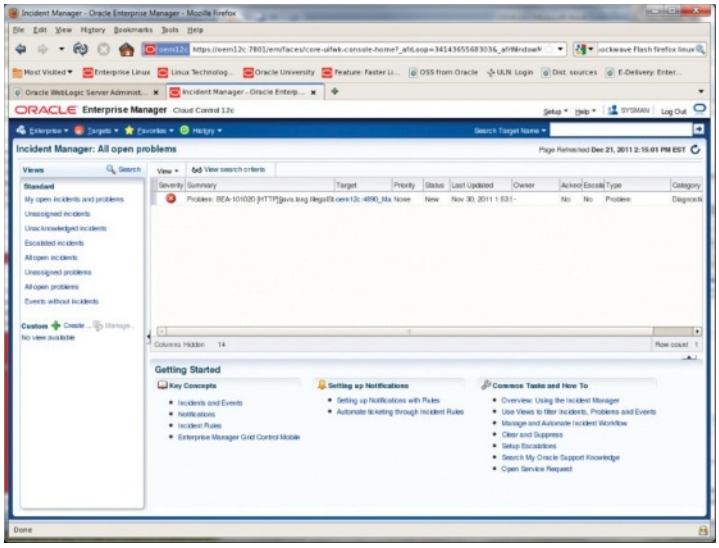


Figure 7: Incidents View

Middleware View

The middleware view would be most appealing to administrators whose primary responsibility is application servers/middleware (Figure 8). The view focuses on middleware targets such as WebLogic.

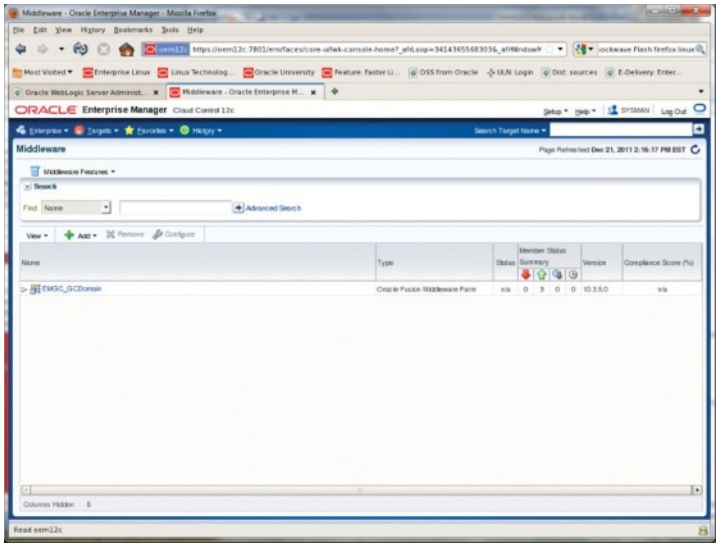


Figure 8: Middleware View

Service Request View

The service request view provides a quick, simple means of managing My Oracle Support service requests and serves as access to the Oracle customer support portal (Figure 9). OEM 12*c* integrates closely with MOS, as detailed later.

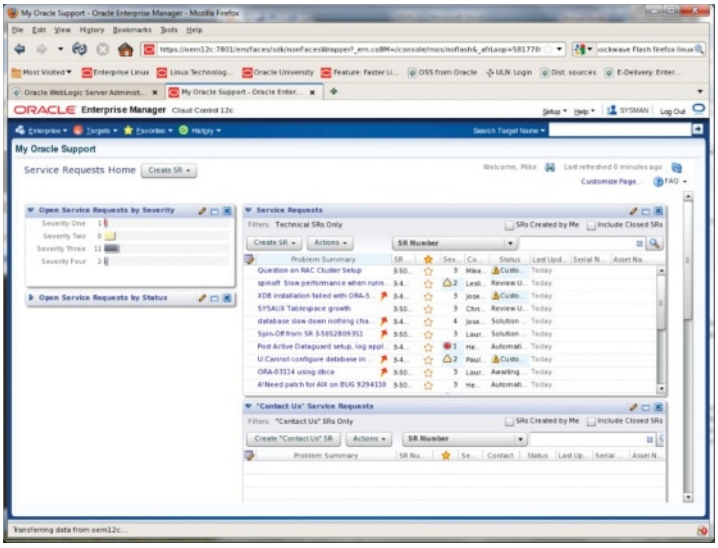


Figure 9: Service Request View

Services View

The services view focuses on enterprisewide services that are subject to our internal service level agreements (Figure 10). The view shows all configured services and service health. This provides managers and executives with insight into system performance at varying service levels and monitors adherence to selected performance based on service-level agreements.

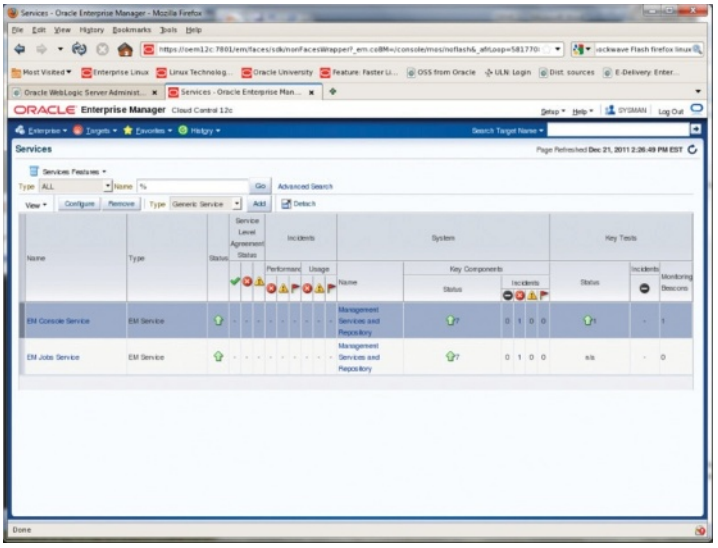


Figure 10: Services View

Global Search Improvements

The presentation, search and filtering functionalities have all been significantly improved in OEM 12*c*. The search dropdown menu gives several search options including Target Name, Knowledge Base, Knowledge Base Archives, Bug Database, Communities, Documentation, Sun System Handbook and All Knowledge (Figure 11).

continued on page 8

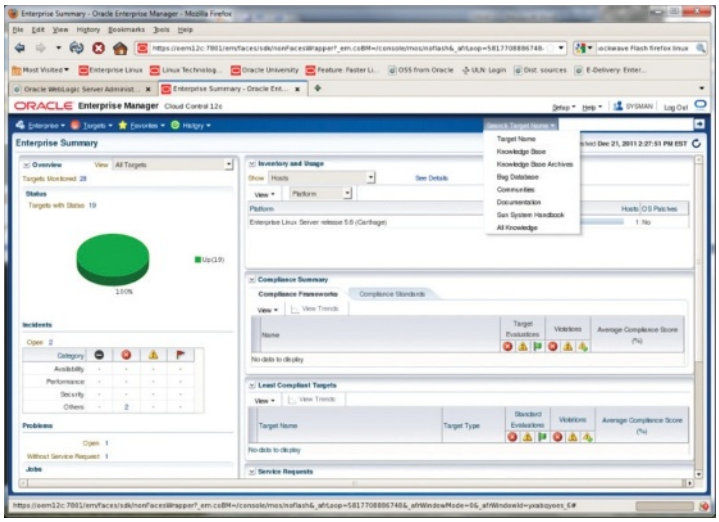


Figure 11: Search and Filter Functionalities

Menu Options

Enterprise Menu

The enterprise menu is a new addition to Oracle Enterprise Manager (Figure 12). This menu option includes navigation to the Summary page, Monitoring, Job, Reports, Configuration Compliance, Provisioning and Patching, Quality Management and My Oracle Support.

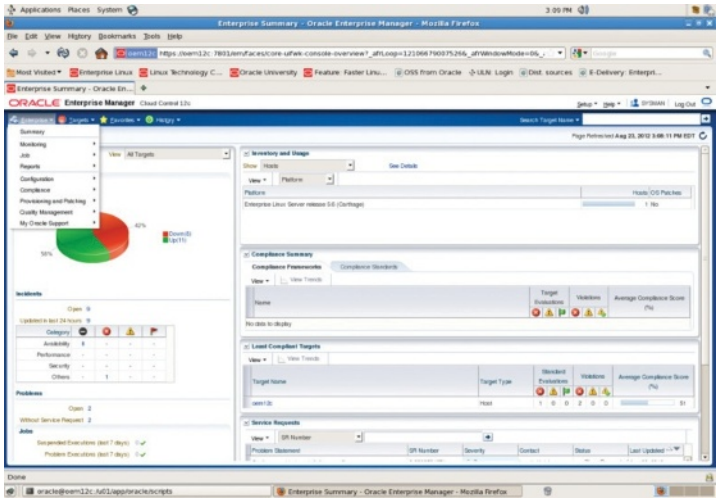


Figure 12: The Enterprise Menu

Summary

The summary option will take you directly to the target summary screen.

Monitoring

The monitoring menu item within the enterprise menu will allow fast navigation to the Incident Manager, Logs, Blackouts, Corrective Actions, Metric Extensions, Monitoring Templates, Support Workbench and Template Collections (Figure 13).

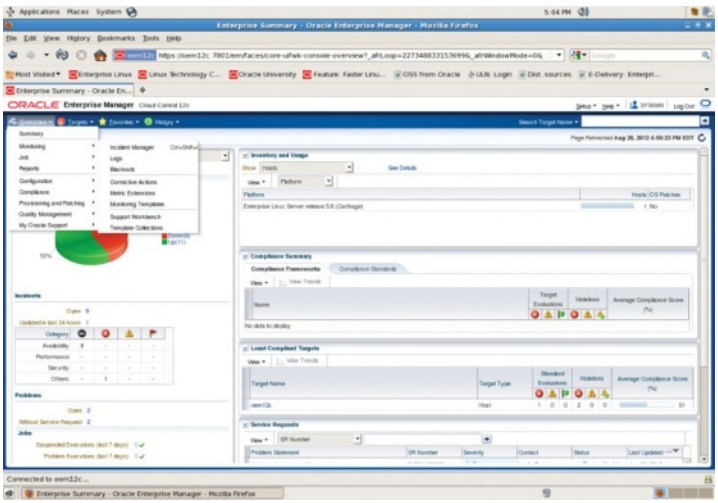


Figure 13: Monitoring Menu

Job

The job menu item within the enterprise menu allows navigation to the Oracle Enterprise Manager Jobs Activity and Jobs Library (Figure 14).

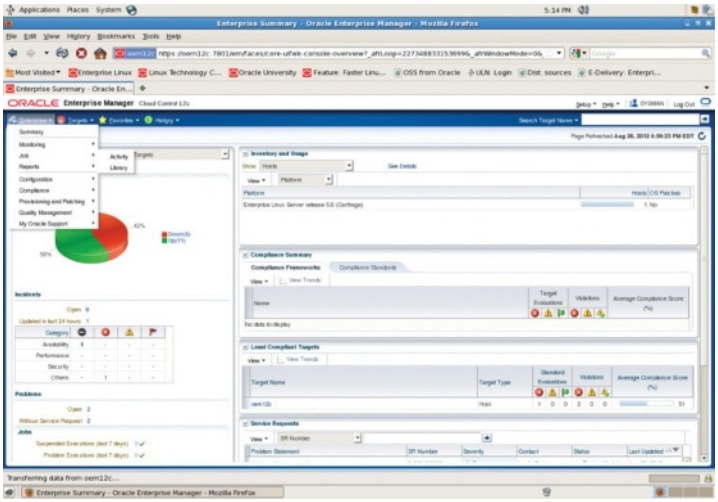


Figure 14: Job Menu

Reports

The reports menu item within the enterprise menu will navigate to the Information Publisher Reports or the BI Publisher Enterprise Reports (Figure 15). The Information Publisher Reports are much like OEM Grid Control Reports, while the BI Publisher Reports are new for Oracle Enterprise Manager 12*c*. * *See Reporting Enhancements*

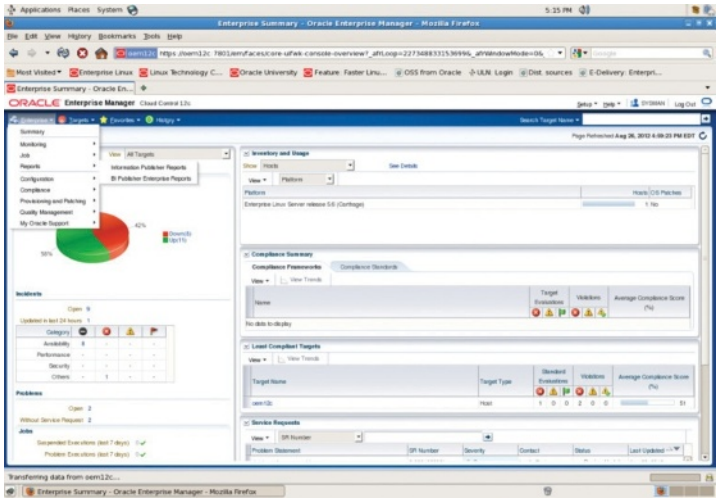


Figure 15: Reports Menu

Configuration

The configuration menu item within the enterprise menu gives quick navigation to the configuration management capabilities of OEM 12*c* such as Comparison of Configurations, Configuration Inventory and Configuration History (Figure 16).

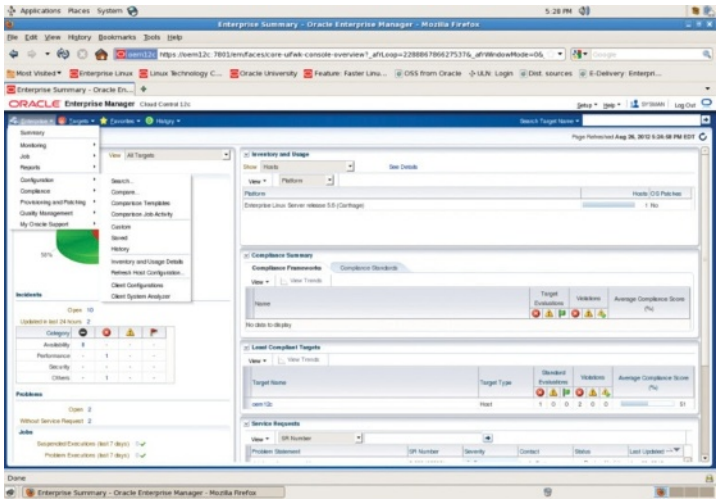


Figure 16: Configuration Menu

Compliance

The compliance menu item within the enterprise menu provides quick navigation to the new compliance management capabilities of Oracle Enterprise Manager 12*c* (Figure 17). ** See Compliance Management and Reporting*

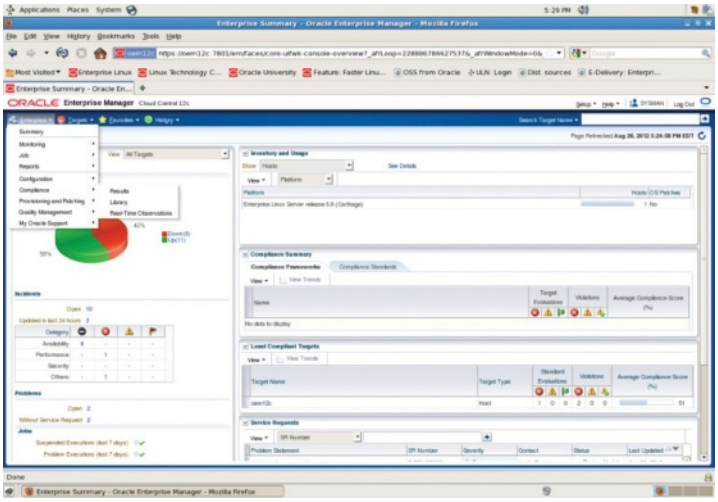


Figure 17: Compliance Menu

Provisioning and Patching

The provisioning and patching menu item on the enterprise menu provides fast access to the provisioning capabilities of Oracle Enterprise Manager 12*c* Cloud Control (Figure 18). Provisioning capabilities include Bare Metal, Database, Middleware and Patches and Updates.

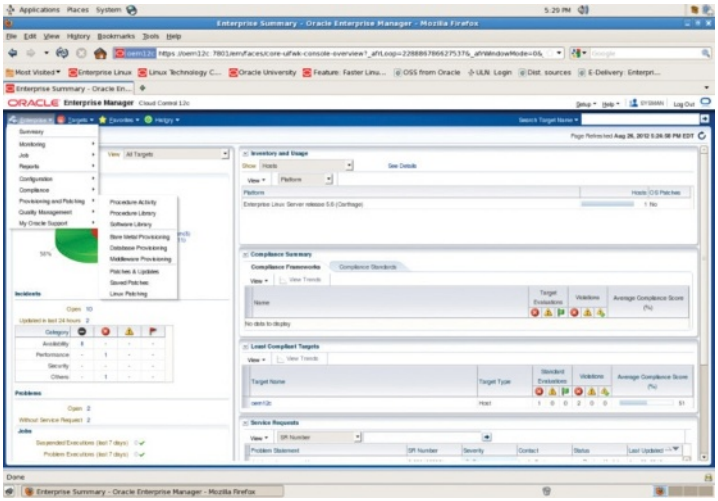


Figure 18: Provisioning and Patching Menu

Quality Management

The quality management menu item in the enterprise menu is for the components of Oracle Enterprise Manager 12*c* to improve the management of your environment (Figure 19). This includes quality of the deployed components through interaction with Oracle Database Real Application Testing, which includes SQL Performance Analyzer and Database Replay. It also includes the ability to mask data in your development and test environments to protect identifiable data through the Data Masking Pack. ** See Data Masking*

continued on page 10

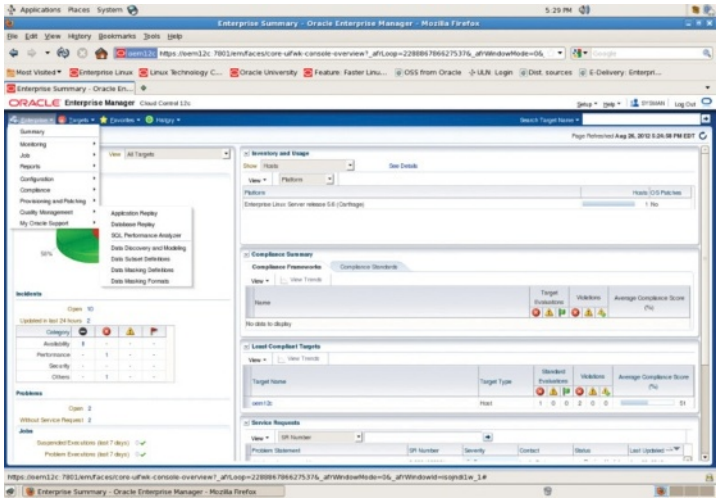


Figure 19: Quality Management Menu

My Oracle Support

The My Oracle Support menu item will navigate Service Requests, My Oracle Support Knowledge Base, My Oracle Support Certification and My Oracle Support Community via My Oracle Support Integration provided in OEM 12*c* (Figure 20). **See My Oracle Support Integration*

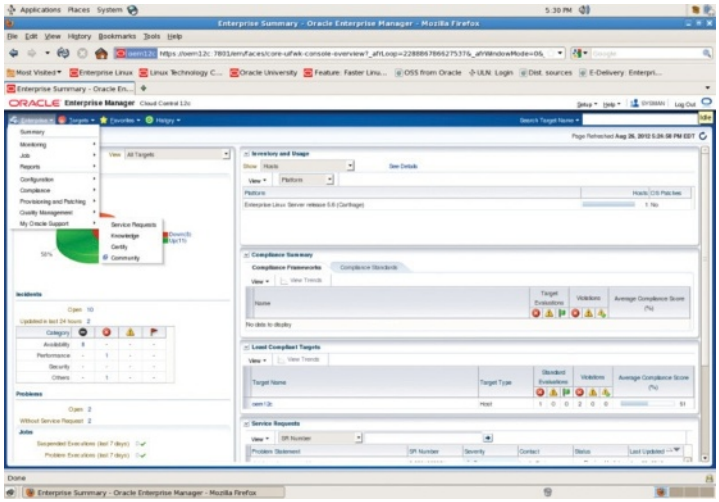


Figure 20: My Oracle Support Menu

Targets Menu

Similar to OEM Grid Control, the targets menu in OEM 12*c* is focused on the monitored targets and offers user-friendly, direct access (Figure 21). In OEM Grid Control, the Targets tab, by default, puts the user in the hosts target list, while in OEM 12*c*, selecting from the Targets menu allowing direct access to the target list. Selections are similar the OEM Grid Tabs available once navigated to the Target tab.

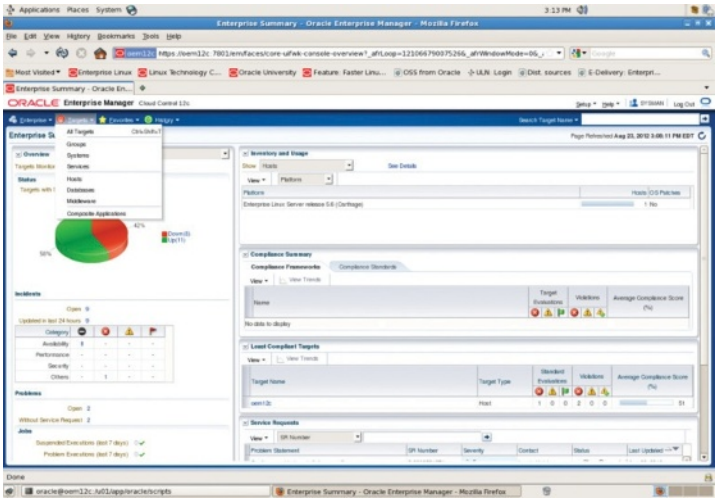


Figure 21: Targets Menu

Favorites Menu

The favorites menu allows each user to designate screens and pages that are used the most or connect the administrator with problem systems for monitoring (Figure 22). The functionality is similar to the Bookmarks/ Favorites feature of any web browser. Favorites contain links providing direct navigation without the need to navigate through the application. OEM Favorites can be updated or removed like browser bookmarks.

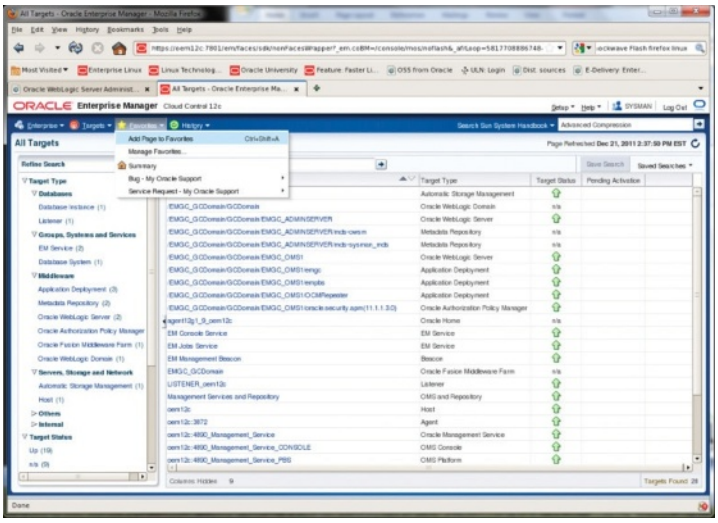


Figure 22: Favorites Menu

Management via Plug-ins

Prior versions of Oracle Enterprise Manager Grid Control had target management capabilities built in; however, the concept of management plug-ins allowed OEM to extend its capability to manage targets that weren't originally managed by OEM Grid Control. With Oracle Enterprise Manager 12*c*, all target monitoring and management capability are provided via a plug-in (Figure 23). This underlying framework change improves OEM 12*c* as it allows a plug-in (target management capability) to be installed or upgraded without having to patch or upgrade Oracle Enterprise Manager.

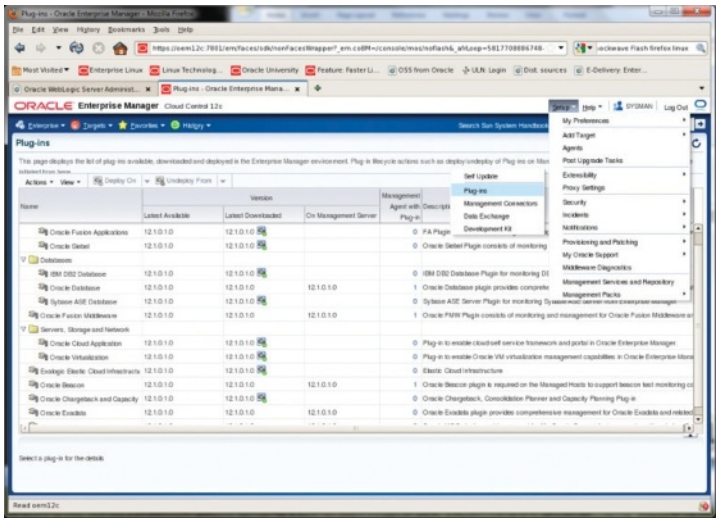


Figure 23: Management via Plug-Ins

Data Masking

Reversible Data Masking

The data masking capability in Oracle Enterprise Manager Grid Control has been improved. Data masking allows organizations with sensitive data to utilize the production data in their development and test environments. In OEM Grid Control, the user could use data masking to create test environments using production data and hide data to keep the sensitive information from being read or understood, therefore protecting it. Oracle Enterprise Manager 12*c* has improved this capability by allowing the data masking process to be reversed. OEM 12*c* masks the data using a format provided by the use in the form of a regular expression. This allows the unmasking, or reversal, of the database so the data can be reverted back to the original.

Data Masking Integration with Real Application Testing

Real Application Testing is a feature that can capture a real production database workload and replay it in another environment. This permits a full database load and all database transaction activity to be tested with a true database load prior to being implemented. OEM 12*c* integrates Data Masking with Real Application Testing.

Application Data Model Support for Data Masking

An Application Data Model (ADM) is now available for certain packaged applications. The ADM can serve as a knowledge base containing sensitive column and data relationships for data masking. For applications with ADM availability, the user can use the ADM to identify which columns contain sensitive information and need to be masked.

Data Masking Format Library

Data Masking Format Library provides formats for different data elements that need to be masked (Figure 24).

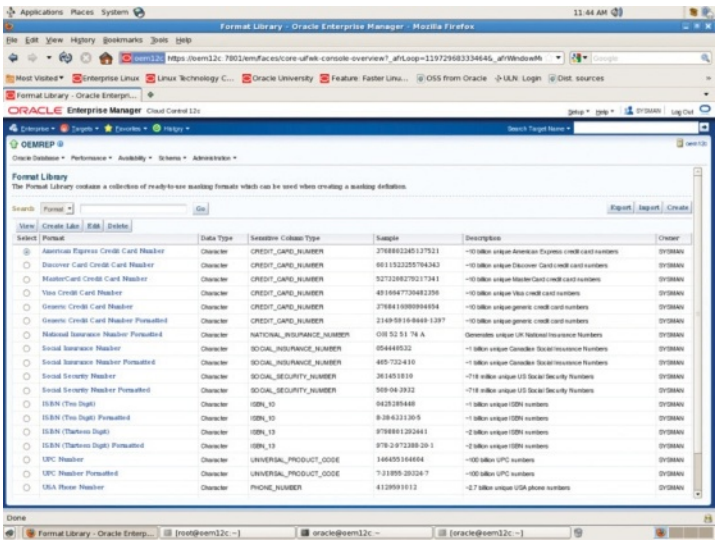


Figure 24: Data Masking Format Library

My Oracle Support Integration

Oracle Enterprise Manager 12*c* Cloud Control provides integration with My Oracle Support to create, monitor and update support service requests and relate them to the configurations within OEM 12*c* (Figure 25). This greatly simplifies the management of service requests, as it allows the monitoring and management activities within the same console.

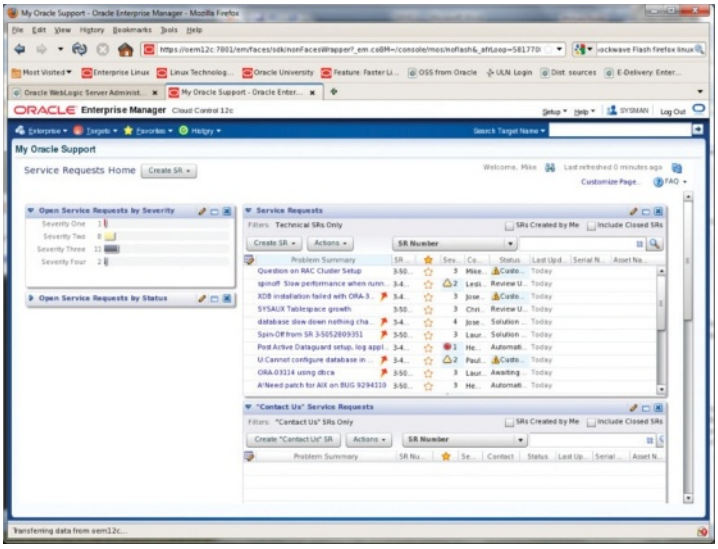


Figure 25: My Oracle Support

Reporting Enhancements

Oracle Enterprise Manager Reporting Enhancements are through the integration with Oracle Business Intelligence (BI) Publisher, which allows for the creation of highly formatted documents and the ability to develop/design reports using known desktop products.

Information Publisher Reports

The familiar OEM Grid Control Reports are still in Oracle Enterprise Manager 12*c* under the Information Publisher Reports (Figure 26).

continued on page 12

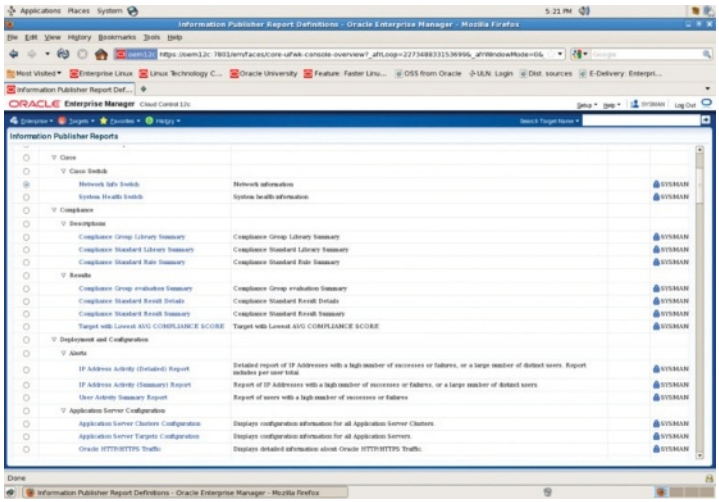


Figure 26: Information Publisher Reports

BI Publisher Reports

This is a feature that must be configured with Oracle Enterprise Manager 12*c* Cloud Control, as it is not configured by default. The Business Intelligence Publisher is an Oracle product that provides a powerful reporting tool for business intelligence environments. With Oracle Enterprise Manager 12*c*, the ability to integrate and utilize BI Publisher with OEM for reporting has been added. The only limitation at this time is that BI Publisher 11.1.1.5.0 is the only one that can be utilized. The BI Publisher 11.1.1.5.0 can be downloaded from the Oracle Technology Network at http://www.oracle.com/technetwork/oem/grid-control/downloads/index.html.

Database

Auto Discovery

One of the advantages of the new Oracle Enterprise Manager 12*c* is target discovery, where targets can be discovered automatically. OEM 12*c* Exadata server management has been improved through auto discovery of Exadata targets and simplified performance diagnostics more specific to Exadata.

Backup and Recovery

OEM 12*c* has also provided improvements in backup and recovery operations by allowing the central management of database and file system backups as well as uniform settings for both database and file system backups.

Emergency Performance

Oracle Enterprise Manager 12*c* introduces the use of the memory access mode to connect to a slow database. This allows the database administrator to troubleshoot the root cause of the problem. The memory access mode bypasses the SQL information retrieval layer and allows the reading of performance statistics directly from the SGA of the database.

Compare Period Advisor

Oracle Enterprise Manager can now compare the performance of a database over two different time ranges. OEM 12*c* analyzes changes in performance, workload, configuration and hardware and will highlight any changes found between the two provided time periods.

Template Integration

Monitoring templates and compliance standard settings and applying to targets and target groups has been streamlined.

Blackout Enhancements

Oracle Enterprise Manager 12*c* offers a consolidated blackout information summary within the general region of a target's home page.

ASM Clusters as a Target

An ASM Cluster in OEM 12*c* allows the management of an ASM cluster as a single target.

What's New?

This section is all about what is new in Oracle Enterprise Manager 12*c*, detailing improvements and functionality repackaged and improved in such a way that it appears anew.

Metric Extensions

Metric extensions allow full-fledged metrics to be created on targets. These are created and managed from the metrics extensions page as well as able to be export and imported between environments (Figure 27).

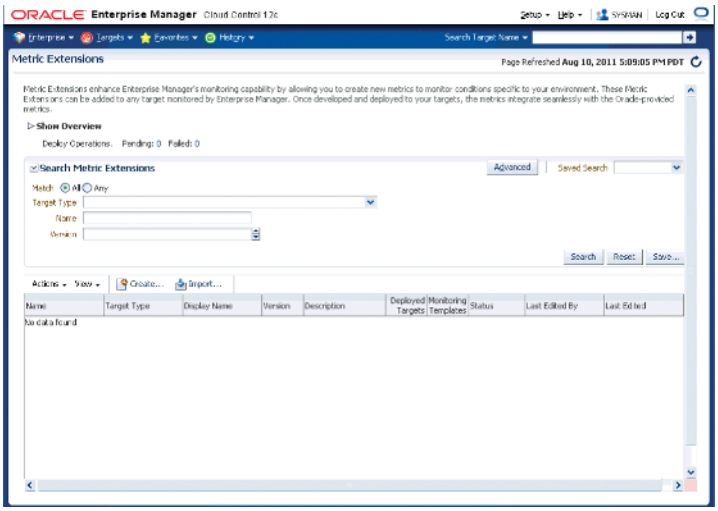


Figure 27: Metric Extensions

How do metric extensions really differ from user-defined metrics used in Grid Control?

- User-defined metrics (UDM) collect values through executing OS scripts or SQL statements. This imposes several limitations:
  - UDM custom scripts/other files require file(s) to be manually placed on each target's file system.
  - User-defined metrics collect multiple data pieces using multiple user-defined metrics and cannot refer to two data points from two metrics for alert generation, as data is collected separately.
  - User-defined metrics only use the OS script and SQL execution, while metric extensions can use protocols such as SNMP and JMX.
  - Only user-defined metrics are allowed for host targets and SQL for database targets. No other target types are allowed, such as WebLogic.

Access

Oracle Enterprise Manager 12*c* greatly improves overall security and access control. It incorporates fine-grained privileges through the introduction of 200 privileges from targets, objects and resources as well as new roles. It also

brings in administration groups, allowing many targets to be managed as if it they were a single object. This allows a single setting change to the administration group to apply to all targets within that group. This simplifies the ability to manage numerous targets by standardizing, managing and monitoring settings for targets.

Database

Database Creation

The new OEM wizard for database creates from within a central location using a GUI-like tool without having to log in directly to the server. This function will streamline the creation of new databases and help improve security on database servers by eliminating the need to log onto the servers to create a new database. This new database creation wizard, much like the database creation assistant, has the ability to create new single instance as well as Real Application Cluster databases.

Database Upgrade

Oracle Enterprise Manager 12*c* gives the ability to upgrade databases from a single console for single instance as well as RAC databases. This can help simplify the database upgrade process as well as make the management easier and lessen the time required to perform database upgrades.

Cloud Policies

What are policies? Policies are the rules in which actions are taken to correct things under defined situations. In OEM 12*c*, a user can define policies or use predefined policies already created. Predefined policies cannot be updated; however, a new policy can be created using the predefined policy as a starting point, which can then be modified. Policies can be created in a hierarchy where a policy at a higher level inherits the properties of the policy at the lower level. In addition to the policy levels, there are two types of policies: performance policies and schedule-based policies. Performance policies are based on the performance metrics of a target, while schedule policies are based on a defined schedule.

For more information and the view the cloud policies, follow these steps:

1. Log in to Enterprise Manager 12*c* Cloud Control.
2. Select the "Cloud" option from the grid menu and then "Select Policies."
3. The policy home page will appear with a list of policies and administration privileges.

Compliance Management and Reporting

Compliance management was introduced with Oracle Enterprise Manager Grid Control, where standard best practices were measured for targets (Figure 28). The best practices cover vulnerability and configuration where there were critical, warning and informational levels.

Oracle Enterprise Manager 12*c* improved upon this with a framework that allows the best practices to be based on business best practices for security, configuration and storage. This framework will also make recommendations for bringing the targets into compliance. OEM 12*c* also now has database configuration data to be managed within the framework. The framework allows for industry standard compliance such as PCI, PCI Requirement 10, PCI 10.5, etc. The standards are evaluated against the compliance standard for the target in OEM and will assign a compliance standard report a score.

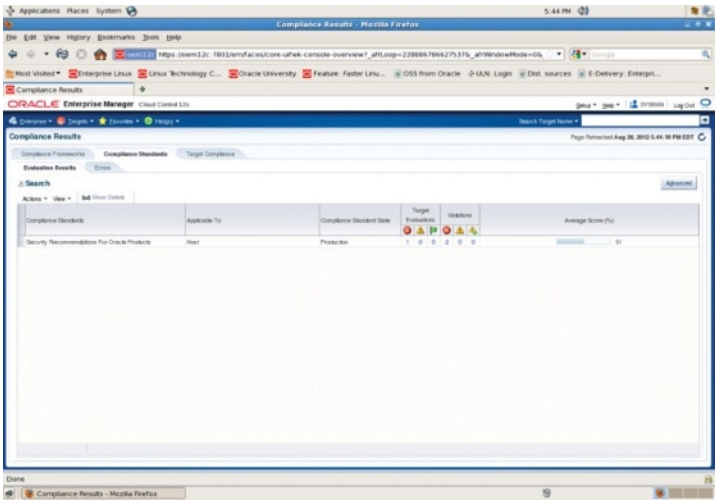


Figure 28: Compliance Management and Reporting

Consolidation Planner

In order to stay competitive, businesses must have the ability to grow and adapt to the increasing amounts of data. Consolidations include: floor space, power, cooling and management. Management needs to include security patching, upgrades, operating system and other software patches. Organizations have come to realize that, in many cases, servers are utilized only partially and, therefore, valuable resources are wasted. Consolidation can help an organization utilize its resources more efficiently, eliminate excess resources in the data center and expand capacity.

This Oracle Enterprise Manager 12*c* permits the consolidation of servers, such as Exadata. The consolidation planner leverages data collected from server targets managed in OEM Cloud Control and factors in business and technical requirements to help you determine the consolidation options.

Incident Manager

Incident manager is a new function offered in OEM 12*c*. Incident manager provides a view and interface to work with incidents in the environment. When an issue is identified by Oracle software, they become incidents recorded in the Automatic Diagnostic Repository. The OEM 12*c* interface manages incidents including the current status, comments, etc.

continued on page 14

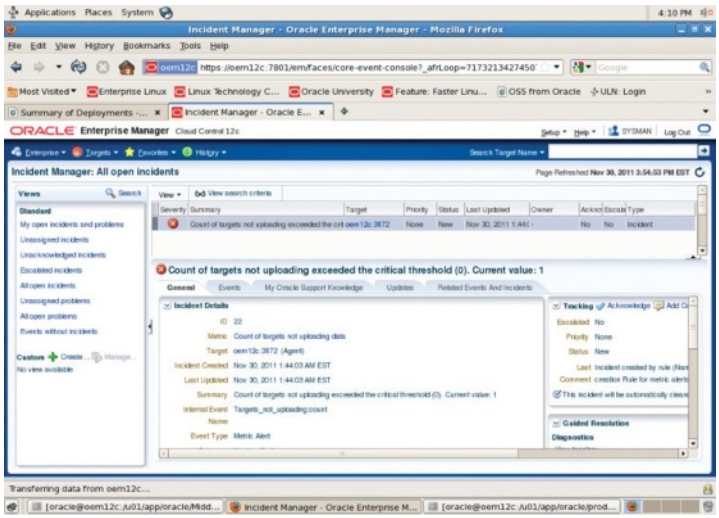


Figure 29: Incident Manager

Incidents

As illustrated in Figure 29, this is an open incident that is not assigned. The user can select the incident to view its details in the lower portion of the screen. This area will permit the user to acknowledge the incident, add comments and manage the incident, which includes assigning the incident, prioritizing it and updating the status.

There are several alternate incident views available, such as unassigned incidents, unacknowledged incidents and escalated incidents.

Incidents vs. Problems

In addition to incidents, OEM 12*c* also recognizes problems, which, in an ITIL framework, are recurring incidents that become a problem, or known issues. Select the "All Open Problems" to display a list of open problems, and then select a problem in order to view the details in the lower pane of the screen (Figure 30).

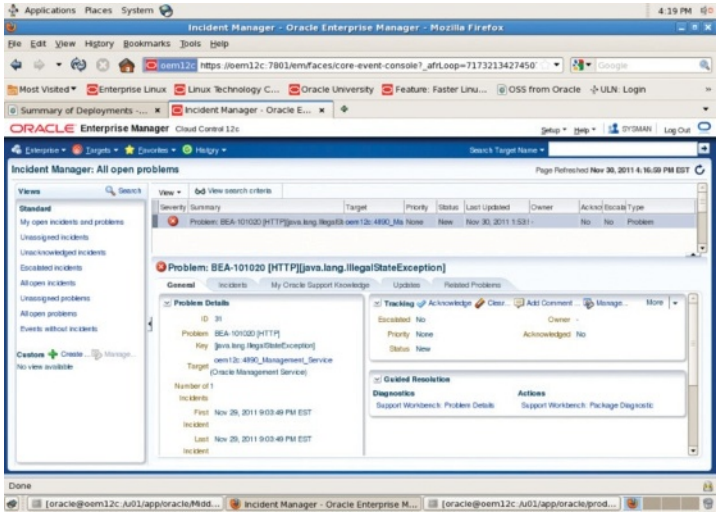


Figure 30: Incident Manager: All Open Problems

In this lower pane, like with incidents, we can manage the problem, but we can also see the related incidents. This area will allow you to acknowledge the problem, add comments and manage the problem, which includes assigning the problem, giving it a priority and updating the status.

Chargeback Administration

When businesses create shared resources, the challenge becomes how to allocate costs to the users for those resources, especially when shared resources scale up and down according to workload needs over time. The challenge is greater when shared resources scale up and down according to workload needs and shift often and vary greatly over time. Thus, the ability to quantify the resource usage and the ability to charge for consumed resources becomes a challenge. Chargeback administration focuses on a utility-based charging model by measuring the consumption of resources so that consumption can be charged based on usage. This model benefits resource consumers by placing them in charge of their resource costs and giving them control over resources they utilize.

OEM collects the metrics (metering, if you will) of resource consumption based on configuration or usage basis. This is done via the targets that OEM sees and measures that target's resource utilization either though the configuration of that resource or the resource usage that target consumes. OEM now gives us the ability to put our own metrics in place to measure resource usage. The metric information allows the measurements of resource usage that can help control resource usage and resource costs. This information is helpful in resource planning.

There are three universal metrics of consumption: CPU, memory and storage. These key resources can have a cost tied to them, for example $1 per GB utilized, $1 per day of CPU utilization and $2 per day of memory utilization per GB. With this, a charge plan can be established, and, with the measurements, a bill can be generated. These universal metrics can be applied to all types of targets, such as a database, a WebLogic Server, etc.

Some Standard Operating Procedures

Add Database Target

Navigate in the OEM Cloud Control to the Database Targets view.

Select "Add" to add a database target.

Enter the host name of the database target and select "Continue." OEM will execute a target discovery process for the host entered.

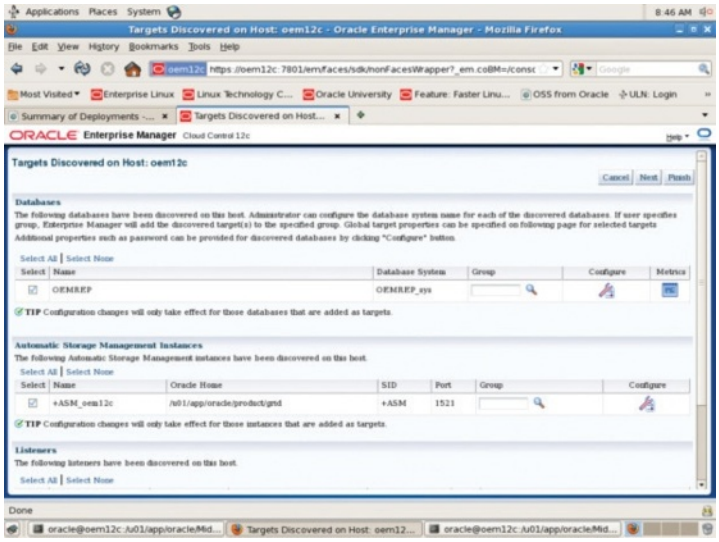


Figure 31: Targets Discovered on Host

Once complete, a list of targets will be displayed. As illustrated in Figure 31, a database and an Automatic Storage Management Instance were found. To add the database, select *Configure*.

Enter the password for the OEM monitor account dbsnmp along with any other connectivity changes (although the proper connection details are usually provided.) Enter the password and select *Test* to ensure the password is correct. Once the password test is completed successfully, select *Next* to continue with the monitoring configuration setup.

Review all the properties and select *OK* to complete the database target monitoring configuration.

Repeat the *Configure* step for any other database or ASM target being added.

Select *Finish* when the target configurations are complete.

A summary of the targets to be added will appear (Figure 32). Select *Save* to add the targets.

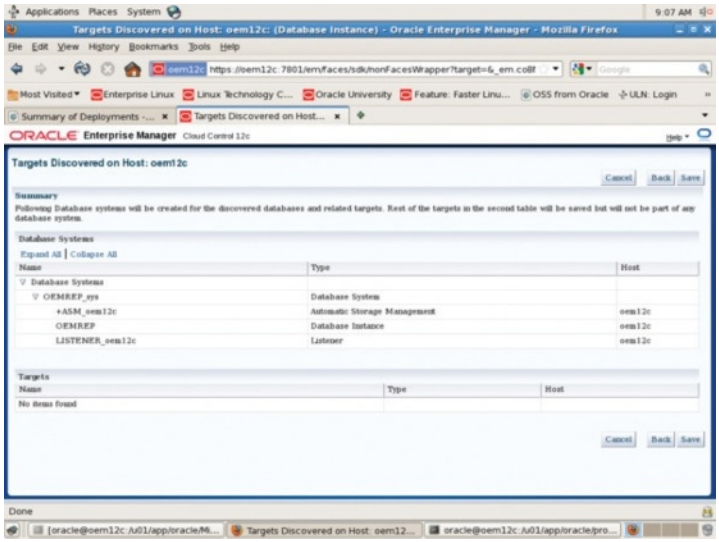


Figure 32: Targets Summary

OEM will perform the target configuration to add the targets.

Once configuration of the targets is complete, OEM will show the configuration results. Select *OK* to complete the process.

Changing Home View

Go to Setup -> My Preferences -> Set Current Page as My Home

*To make the current page a home page, select *Set Current Page as My Home*, otherwise select *Select My Home* (Figure 33)

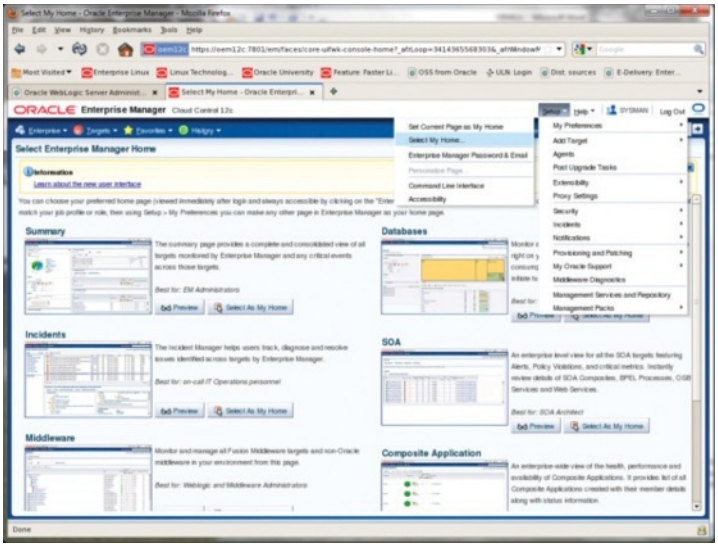


Figure 33: Select My Home

Select a home page.

Once selected, the new home page will appear as the new home page.

Setup My Oracle Support

Go to Setup -> My Oracle Support -> Settings

Enter My Oracle Support credentials then select *Go*.

Personalize, manage users, etc., using My Oracle Support.

About the Author

Michael Messina is an Oracle ACE, Oracle certified professional, Oracle RAC administrator certified expert and Exadata implementation specialist with nearly 20 years of experience with Oracle Database. He also authored several chapters in the newest "Oracle Database 11*g* Release 2 Performance and Tuning Tips & Techniques" from Rich Niemiec. A past presenter at Oracle OpenWorld, RMOUG Training Days, IOUG COLLABORATE and many local and regional users groups throughout the United States, Messina's experience includes Oracle and MySQL database administration and implementation, system and infrastructure implementation, systems design and development. He has led several performance improvement, maintenance and implementation projects on large, highly available systems.

Simplify Database Performance Monitoring and Reporting with Foglight for Oracle

No other Oracle database monitoring tool has the power to accelerate problem resolution like Foglight® for Oracle. It provides constant remote database monitoring and correlates performance data from across your technology stack. Foglight for Oracle empowers you to take action immediately and maximize your Oracle database resources.

Read more at **quest.com/foglight-for-oracle/**

The power to do more

Introduction to Oracle Enterprise Manager Command Line Interface

By Ray Smith

Oracle Enterprise Manager Command Line Interface (EMCLI) performs OEM tasks without clicking through the GUI console. This brief paper will introduce general concepts of EMCLI and illustrate its use.

Technical Background

Architecture

Oracle Management Service (OMS) is the heart of OEM 12*c* Cloud Control. The OMS is a J2EE application running on a dedicated Fusion server. An EMCLI extension runs concurrently with the console and upload management services to handle command-line calls to the OMS, acting as a headless OMS. It performs the same tasks but returns text streams instead of pretty pictures. Each EMCLI command is exercised in the EMCLI Java module through the use of verbs.

Verbs

Today there are 333 EMCLI verbs, and Oracle continues to expose more of them as users request them. Each verb is well-documented in Oracle Document 17786-02 and at the command line using this syntax:

```
EMCLI -help
EMCLI -help <verb>
```

You can also refer to the online reference through the setup drop-down in OEM, as shown in Figure 1.

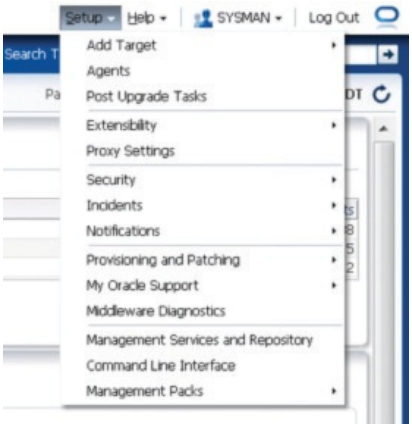


Figure 1

The documentation clearly provides the list of required and optional parameters for each verb, with an example for nearly all verbs. The trick, of course, is turning this knowledge into something useful.

Working with EMCLI

The EMCLI utility is invoked by calling it by name, just like SQLPLUS, EXPDP, RMAN and other Oracle utilities. Unlike sqlplus, EMCLI does not require a subshell[1] when used in a shell script. You simply call it by name like any other UNIX utility (Figures 2 and 3).

```
oracle@myoemserver> emcli setup
Oracle Enterprise Manager Cloud Control 12c Release 12.1.0.1.0.
Copyright (c) 1996, 2012 Oracle Corporation and/or its affiliates. All rights reserved.

Instance Home          : /orabase/Middleware/gc_inst/em/EMGC_OMS1/sysman/emcli/setup/.emcli
EM URL                 : https://myoemserver.corp.dom:7802/em
EM user                : sysman
Trust all certificates : true
```

Figure 2

```
oracle@myoemserver> emcli status
Oracle Enterprise Manager Cloud Control 12c Release 12.1.0.1.0.
Copyright (c) 1996, 2012 Oracle Corporation and/or its affiliates. All rights reserved.

Instance Home          : /orabase/Middleware/gc_inst/em/EMGC_OMS1/sysman/emcli/setup/.emcli
Status                 : Configured
EMCLI Home             : /orabase/Middleware/oms/bin
EMCLI Version          : 12.1.0.1.0
Java Home              : /orabase/Middleware/jdk16/jdk/jre
Java Version           : 1.6.0_24
Log file               : /orabase/Middleware/gc_inst/em/EMGC_OMS1/sysman/emcli/setup/.emcli/.emcli.log
EM URL                 : https://myoemserver.corp.com:7802/em
EM user                : sysman
Auto login             : false
Trust all certificates : true
```

Figure 3

Login and Logout

Basic EMCLI verbs *help*, *setup* and *status* don't require authentication with OMS because they relate to the EMCLI utility itself. Any other commands require a login and, of course, a logout when your session is complete.

```
oracle@myoemserver> emcli login -user=sysman -pass=abc123!
oracle@myoemserver> emcli logout
```

[1] A Shell scripts run SQLPLUS calls in a subshell (<<EOF) with this type of notation:

```
sqlplus system/manager@orcl <<EOF
   SELECT sysdate FROM dual;
exit;
EOF
```

continued on page 18

Targets, Target Types and EMCLI

Targets are the key identifier in OEM. Databases, hosts, listeners, administrators and OEM roles are all target types. Each database name, host name and listener name are unique targets of a type. EMCLI relies heavily on these relationships to determine what to do. Notice that the target types for Oracle products (oracle_database, oracle_listener, f.e.) are very specific.

```
host
j2ee_application
metadata_repository
oracle_apm
oracle_beacon
oracle_database
oracle_dbsys
oracle_emd
oracle_emrep
oracle_em_service

oracle_home
oracle_ias_farm
oracle_listener
oracle_oms
oracle_oms_console
oracle_oms_pbs
rac_database
weblogic_domain
weblogic_j2eeserver
```

Discovery of specific target names can be filtered by target type:

```
oracle@myoemserver> emcli get_targets -target=oracle_database
0               Up              oracle_database             gold
0               Up              oracle_database             bronze.world
0               Down            oracle_database             sandbox_db
```

Target Names and the Discovery Process

In the previous example, you may have noticed that one of the database names (`bronze.world`) included a suffix and the other two did not. This difference is the result of the automated target discovery process that the EM agent performs if you don't manually discover database targets in the 12*c* console.[2]

The automated discovery process runs like this:

1. The agent looks at oratab for Oracle homes.
2. The agent then looks for cluster-ready services that are not present in oratab.
3. For each Oracle home, the agent polls each listener for related database service names either through `lsnrctl` or by parsing the `listener.ora` file.
4. The results of this discovery are saved in the OEM repository exactly as they are found in the listener or the clusterware.

Discovery Consequences

The EMCLI tasks you perform against database targets must use the specific target name, so you need to query the repository (via OMS through EMCLI) for the exact answer. Let's walk through some examples to turn that into knowledge into something you can use.

[2] Host content is discovered by OEM after you add a host agent via the console setup drop-down menu under the heading *Add Targets Manually*. Adding targets using the *Add Non-Host Targets Using Guided Process (Also Adds Related Targets)* follows the discovery process described in this article. You can also control the discovery by following the process behind the *Add Non-Host Targets by Specifying Target Monitoring Properties* radio button. The guided process discovers all targets of a group (the Oracle Database, Listener and Automatic Storage Management group are discovered together) while the individual database, listeners and ASM targets are configured one at a time using the Target Monitoring Properties process.

Example 1: Finding the exact name of a database target

OEM selects a unique name for each registered database target. For reasons beyond the scope of this paper[3], the specific name of a database target may be qualified with the host name (or not) and stored as upper or lowercase without a discernible reason. Even databases created from cloned virtual machines can display different name styles. Manipulating the target database with EMCLI requires an exact match for the target name, so every EMCLI shell scripts must use the exact unique target name from the repository.

Step 1 – Consult the manual

A visit to the documentation is the first step in any EMCLI exercise. Most verbs to query the repository start with a "get." We need specific information about EM targets so we'll use `get_targets`[4] and filter it with *-targets*[5] and *-format* parameters to define our selection.

Step 2 – Select output format

Query results can be returned in one of three predefined layouts or in user-defined layouts where you select the row and column separators. The *pretty* format aligns the column headings with the content, so it makes a good choice for reports. The *script* and *csv* formats are convenient for shell scripts because they are separated by spaces and commas, respectively. The script format is useful with *awk* and *csv* lends itself to the *cut* command, particularly if grep-out the header row.

```
oracle@myoemserver> emcli get_targets -target=oracle_database -
format="name:pretty"
Status
ID          Status          Target Type             Target Name
0           Up              oracle_database         gold
0           Up              oracle_database         bronze.world
0           Down            oracle_database         sandbox_db
```

```
oracle@myoemserver> emcli get_targets -target=oracle_database -
format="name:script"
Status ID          Status          Target Type             Target Name
0                  Up              oracle_database         gold
0                  Up              oracle_database         bronze.world
0                  Down            oracle_database         sandbox_db
```

```
oracle@myoemserver> emcli get_targets        -target=oracle_database \
          -format="name:csv" | grep -v "Status"
0,Up,oracle_database,gold
0,Up,oracle_database,bronze.world
0,Down,oracle_database,sandbox_db
```

```
oracle@myoemserver> emcli get_targets        -target=oracle_database -
format="name:csv" \
 | grep -i "gold" | cut -d, -f4
oracle@myoemserver> gold
```

[3] Database target fact discovery may differ between instances started with spfiles and init.ora files.

[4] This may seem blindingly obvious, but half the EMCLI battle consists of finding the right verb for the job.

[5] You can get a list of all target types in your environment by running *emcli get_targets* without any parameters.

Step 3 – Build that query into a reusable shell script function

You may choose to include this simple query in each of your EMCLI shell scripts, but building it into a scalable script function reduces the maintenance, removes one failure mode from each use (because this module is pre-tested) and increases the readability of your shell scripts. Let's walk through the process.

Step 3A – Handle edge cases

An essential element of any UNIX program[6] is the ability to handle failures quickly and clearly. So the first step in building an industrial strength script from the get_targets snippet is to wrap it in clear messages to the user (or log file) at each step (Figure 4).

```
emcli login -user=sysman -pass="${CONSOLE_PWD}"
echo "Getting the exact target name for $mySID from OMS ..."
if [ `emcli get_targets -targets="oracle_database" | grep -i ${mySID} | wc -l` -gt 0 ]; then
        myTARGET=`emcli get_targets
                        -targets="oracle_database"
                        -format="name:csv" | grep -i ${mySID} | cut -d, -f4`
   echo "${myTARGET}"
else
   echo "Sorry, ${mySID} database is not in the Grid Control repository"
   ExitCleanly
fi
```

Figure 4

Step 3B – Make it into a shell script function

Creation and application of shell script functions for bash and Korn shells consist of using the keyword function, giving the function a name, and wrapping the code inside braces {} as shown in Figure 5.

```
function GetTargetName {
emcli login -user=sysman -pass="${CONSOLE_PWD}"
echo "Getting the exact target name for $mySID from OMS ..."
if [ `emcli get_targets -targets="oracle_database" | grep -i ${mySID} | wc -l` -gt 0 ]; then
        myTARGET=`emcli get_targets
                        -targets="oracle_database"
                        -format="name:csv" | grep -i ${mySID} | cut -d, -f4`
   echo "${myTARGET}"
else
   echo "Sorry, ${mySID} database is not in the Grid Control repository"
   ExitCleanly
fi
}
```

Figure 5

The function is executed inside your shell script simply by calling by name, as you'll see later in this article.

Modularity Within a Script

In the example below, files will be duplicated from a source directory to a destination directory. A list of files is created for each source directory and then the files in that list are copied to the source directory (Figure 6).

```
#  Functions
# ---------------------------------------------------------------------
function CopyFiles {
cd $SOURCE_DIR
ls -1 | grep -i $ORACLE_SID       >$WORKFILE
for thisFILE in `cat $WORKFILE`; do
    SOURCE_FILE=$SOURCE_DIR/$thisFILE
    TARGET_FILE=$TARGET_DIR/$thisFILE
    cp -f $SOURCE_FILE $TARGET_FILE
done
rm -f ${WORKFILE}
}

# ---------------------------------------------------------------------
#  Run-time procedure
# ---------------------------------------------------------------------
SOURCE_DIR=${GOLD_DIR}/rman
TARGET_DIR=${LIVE_DIR}/rman
    CopyFiles

SOURCE_DIR=${GOLD_DIR}/security
TARGET_DIR=${LIVE_DIR}/security
    CopyFiles
```

Figure 6

Step 4 – Store the function in a function library file

Efficient coding practice uses modules, like shell functions, whenever possible. You can extend that practice by saving common functions in function libraries whenever identical code will be used by more than two scripts. *Sourcing*[7] the function library loads the file contents (functions) into memory. The functions can then be called any time during your session.

In addition to the obvious advantage of editing a single copy of the function for every use in your environment, your shell scripts themselves can be significantly simpler as we'll see in the next section.

Example 2: Blackout scripts

OEM blackouts suspend alerting for a specific target during a blackout period. This can be particularly useful during database or host maintenance. The target continues to be monitored, but you won't be paged for a planned outage.

Creating a blackout in the console requires a minimum click-through on six console screens. Fortunately, anything you can define through those screens can also be managed directly from a shell script.

Consider the situation where a training database is refreshed/reset by restoring a cold backup. It happens every night and is scheduled through cron. The entries might consist of a series of calls, first to enable a blackout on the "gold" database in this example, the restore is executed and then blackout is killed off.

```
25 19 * * * /scripts/oem/create_blackout.sh gold
30 19 * * * /scripts/refresh/restore_training_baseline.sh gold
25 20 * * * /scripts/oem/end_blackout.sh gold
```

Step 1 – Consult the manual

Three verbs with very intuitive names are applied to manage blackouts: create_blackout, stop_blackout and delete_blackout. Each of these blackout verbs uses a common parameter: blackout_name.

[6] "The Art of UNIX Programming" by Eric S. Raymond should be required reading for anyone writing code for UNIX/Linux systems. Shell scripts should reflect the transparency, clarity and scalability inherent in Oracle products

[7] A profile file in your home directory is *sourced* whenever you log into a UNIX/Linux host. The contents of the file (paths, aliases and common variable values) are loaded into memory for use within your session. Sourcing a function library from a shell script results in the same behavior.

continued on page 20

Blackouts run as database jobs in your repository database, so this trio of verbs defines, starts, stops and deletes blackout jobs in OEM.

Step 2 – Write the core EMCLI commands The create_blackout verb has the capability to set any of the values you can set from the OEM console, but a minimum of four values should be provided. Those parameters are described in Table 1. Also, see the code in Figure 7.

Parameter	Description
-name	Blackouts are managed by the Oracle Management Service and run as scheduled jobs. Each job requires a distinct name, including blackouts. In this case, the blackout name ${BO_NAME} will be passed in from the shell script.
-add_targets	Blackouts can consist of any combination of OEM targets, including oracle_ database and oracle_listener. In this example, we'll black out a single database base on the value of ${myTARGET} passed in from the shell script.
-schedule	The combinations available for the schedule parameter are beyond the scope of this paper, but this example sets a three-hour duration for a blackout, invoked immediately (the null between the duration and 360 above) in the author's home time zone.[8]
-reason	Pass a string to explain the reason for your blackout.[9]

Table 1

```
emcli create_blackout -name="${BO_NAME}" \
 -add_targets=${myTARGET}:oracle_database \
 -schedule="duration::360;tzinfo:specified;tzregion:America/Los_Angeles" \
 -reason="Scripted blackout for maintenance or refresh"
```

Figure 7

Step 3 – Functionalize the EMCLI code

Robust shell scripts handle expected situations during execution, and the create_blackout function is no exception. (See Figure 8)

Issue 1 – Determining the exact target name

Each blackout is for a specific target or groups of targets, so we'll apply the GetTargetNames function we create earlier to set the name for us.

Issue 2 – Existing blackouts

When you create a blackout using a formula in a shell script, you have the potential for blackout conflicts. When this conflict arises, we can either exit politely and hope that enough time remains on the last blackout or rebuild the blackout to ensure coverage.

Issue 3 – Trust

Echoing status to a log or to the screen provides the assurance that the script performed as expected.

[8] Default settings for the *schedule* parameter include an immediate start and an unspecified duration.

[9] Best practice suggests internal documentation whenever possible. You aren't required to include a reason.

```
function CreateBlackout {

GetTargetName

echo "Creating blackout named '${BO_NAME}' ..."

if [ `emcli get_blackouts | grep ${BO_NAME} | wc -l` -gt 0 ]; then
   echo "Found an existing blackout named ${BO_NAME}"
   echo "That blackout will be stopped and deleted prior to starting the new one"
   emcli stop_blackout -name="${BO_NAME}"
   emcli delete_blackout -name="${BO_NAME}"
fi

emcli create_blackout -name="${BO_NAME}" \
 -add_targets=${myTARGET}:oracle_database \
 -schedule="duration::360;tzinfo:specified;tzregion:America/Los_Angeles" \
 -reason="Scripted blackout for maintenance or refresh"

sleep 5
echo "Getting blackout information for '${BO_NAME}' ..."
emcli get_blackout_details -name="${BO_NAME}"
}
```

Figure 8

Step 4 – Build the shell script to create the blackout (Figure 9)

```
#!/bin/ksh
###########################################################################
#  File:       emcli_start_blackout.ksh
#  Purpose:    Create and initiate an OEM blackout
#  Parameters: Database name
###########################################################################
# Source the function library
. /scripts/oem/emcli_functions.lib

# Verify that SID was passed on command line
if [ ${#1} -eq 0 ]; then
   read thisSID?"Enter the name of the database to be blacked out:  "
else
   thisSID=`print $@ | awk '{print$NF}' | tr '[A-Z]' '[a-z]'`
fi

BO_NAME=scripted_blackout_${thisSID}

CreateBlackout

ExitCleanly
```

Figure 9

Process

1. Both of the functions we've created are stored in a common file called /scripts/oem/EMCLI_functions.lib in this example. Sourcing the library loads the functions into your environment for use.
2. This script can either be run interactively or using cron. The database name *must* be passed to the script either (in this example) as a command line variable or through response to a prompt.[10] The *if* statement required to fail-out when ${thisSID} is null has been take-n out of this example for clarity — your final script should include it.
3. The variable BO_NAME[11] is set based on the value of thisSID.
4. The function CreateBlackout is invoked.
5. The ExitCleanly function consists of temporary file cleanup and logging out of your EMCLI session.

[10] Apply your own method for ensuring scalability from a single set of scripts. Avoid hard coding whenever possible.

[11] Some variable names suggest themselves.

Step 5 – Ending the blackout

Four parameters were required to create a blackout, but only the name is required to stop and delete the blackout. Blackouts must be stopped before being deleted, because the blackout is an OMS repository job.

The wrapper script for this function is identical to the `create_blackout` script, but this script submits the following function `EndBlockout` for `CreateBlackout` in the start script (See Figure 10).

```
function EndBlackout {

GetTargetName

echo "Stopping blackout '${BO_NAME}' ..."
emcli stop_blackout -name="${BO_NAME}"

echo "Deleting blackout '${BO_NAME}' ..."
emcli delete_blackout -name="${BO_NAME}"

}
```

Figure 10

Extensibility

The crontab entry illustrated earlier relied on best-guess timing. The blackout creation script was scheduled to run a few minutes before the restore script kicks off, and the script to end the blackout was scheduled to run about an hour later.

Placing the CreateBlackout and EndBlackout scripts in a function library allows those two tasks to be integrated within any of your shell scripts instead of separate script executions. Use the same technique illustrated in the `emcli_start_blackout.ksh` script to source the function library at the beginning of the restore script, and call those two functions from inside the script. Apply this to any script that requires OEM blackout management for seamless, integrated operations.

Example 3: Handling Planned Outages

Scheduled maintenance events provide another opportunity to schedule blackouts in a slightly different manner, using information from the EM agent to dynamically generate script.

The Right Tool for the Job

It's very tempting to use EMCLI as the one-size-fits-all tool once you become familiar with its features, but its flexibility starts to shine when you combine EMCLI with other data sources.[12]

The EM agent is controlled and queried through the emctl utility. You can quickly determine the targets monitored by the local agent using emctl.

```
oracle@demohost> emctl config agent gettargets
[agent12g1_1_demohost, oracle_home]
[BRONZE.WORLD, oracle_database]
[demohost.com, host]
[demohost.com:3872, oracle_emd]
[LSNRBRONZE_demohost.com, oracle_listener]
[LSNRSILVER_demohost.com, oracle_listener]
[OraDb11g_home1_2_demohost, oracle_home]
[SILVER, oracle_database]
```

You can quickly turn those results into a list of database and listener targets to blackout with simple grep statement for the oracle_% target types and apply a little creative "cutting" to create two target lists for the databases and listeners.

```
oracle@demohost> emctl config agent listtargets | grep oracle_database \
                 | cut –d[ -f2 | cut –d, –f1 > databases.lst

oracle@demohost> cat databases.lst
 BRONZE.WORLD
 SILVER

oracle@demohost> emctl config agent listtargets | grep oracle_listener \
                 | cut –d[ -f2 | cut –d, –f1 > listeners.lst

oracle@demohost> cat listeners.lst
 LSNRBRONZE_demohost.com
 LSNRSILVER_demohost.com
```

The argfile Verb

Batch or list processing for multiple consecutive EMCLI commands is captured in the *argfile* verb. The syntax is quite simple:

```
oracle@demohost> emcli argfile /fully_qualified/path/filename.lst
```

All of the verbs listed in the argfile are executed in a single connection to the OMS server and process very efficiently. Let's walk through the process of generating and executing two argfiles to start and stop blackouts for the database and listener targets from the example above.[13]

```
touch blackout_start.cmd
touch blackout_end.cmd

export BONAME=GENERATED_BLACKOUT
echo "create_blackout -name=${BONAME} " >>blackout_start.cmd
for thisDB in `cat databases.lst`; do
 echo " -add_targets=${thisDB}:oracle_database " >>blackout_start.cmd
done
for thisLSNR in `cat listeners.lst`; do
 echo " -add_targets=${thisLSNR}:oracle_listener " >>blackout_start.cmd
done
echo " -schedule=duration::360;tzinfo:specified;tzregion:America/Chicago "
>>blackout_start.cmd
echo " -reason=Scripted blackout for maintenance" >>blackout_start.cmd
echo "stop_blackout -name=${BONAME}" >>blackout_end.cmd
echo "delete_blackout -name=${BONAME}" >>blackout_end.cmd
```

The loops in that short script generate the argfiles `blackout_start.txt` and `blackout_end.txt`.

[12] The argfile example in this article uses techniques we've already learned to illustrate how argfiles can be used. Blackouts for all targets on a host can be created using the *propagate_targets* parameter. Consult the documentation for specific syntax.

continued on page 22

```
oracle@demohost> cat blackout_start.cmd
 create_blackout -name= GENERATED_BLACKOUT \
 -add-targets=BRONZE.WORLD:oracle_database \
 -add-targets=SILVER:oracle_database \
 -add-targets= LSNRBRONZE_demohost.com:oracle_listener \
 -add-targets= LSNRSILVER_demohost.com:oracle_listener \
 -schedule=duration::360;tzinfo:specified;tzregion:America/Chicago \
 -reason=Scripted blackout for maintenance

oracle@demohost> cat blackout_end.cmd
 stop_blackout -name= GENERATED_BLACKOUT
 delete_blackout -name= GENERATED_BLACKOUT
```

Executing these blocks consists of invoking emcli with the argfile command. Notice that each command starts with a verb and not "emcli."

```
oracle@demohost> emcli argfile blackout_start.txt

< Do some work on the host>

oracle@demohost> emcli argfile blackout_end.txt
```

Summary

OEM 12*c* command line interface provides a convenient means of automating tasks on the command line or inside shell scripts. The first step in implementing EMCLI in your environment is to download a copy of the reference document. A comprehensive list of verbs is also included online and can be accessed through the console. The examples provided here provide a starting point.

My Oracle Support now features community interaction for challenges you encounter. Click on the *Communities* tab to subscribe to the Enterprise Manager communities (Figure 11).

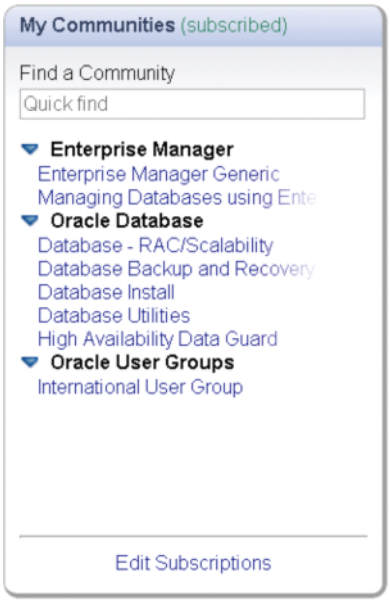


Figure 11

About the Author

Ray Smith is actively engaged as a volunteer for IOUG. He has served on COLLABORATE and Oracle OpenWorld, and is on the *SELECT Journal* Editorial Board. He currently works as a senior Oracle DBA/technologist for Portland General Electric in Portland, Ore. He is also an Oracle ACE.

USERS GROUP CALENDAR

For the most updated calendar, please visit www.ioug.org

IOUG WEBINARS:

Did you miss an event? Watch previous IOUG webcasts on-demand here: http://www.ioug.org/p/cm/ld/fid=153.

APRIL 2013

April 7-11
COLLABORATE '13
Denver
Event URL:
http://collaborate13.ioug.org/p/cm/ld/fid=112

SEPTEMBER 2013

Sept. 22-26
Oracle OpenWorld
San Francisco
Event URL:
http://www.oracle.com/openworld

Oracle Enterprise Manager: Next Generation Training

COLLABORATE 13 – IOUG Forum is the conference to attend for the latest and greatest in user-driven Oracle education and training. Check out a sneak preview of some of the top IOUG sessions on Oracle Enterprise Manager:

Monday, April 8: Performance Tuning your DB Cloud in OEM 12*c* Cloud Control - 360 Degrees, presented by Tariq Farooq

Monday, April 8: EM12*c* – Less Known Features and Techniques, presented by Kellyn Pot'Vin

Tuesday, April 9: Simplifying Application Deployment in Cloud using Virtual assemblies and EM 12*c*, presented by Honglin Su, Kai Yu

Check out a full preview of all IOUG offerings for Oracle users of OEM today at **collaborate13.ioug.org/oem** and register to reserve your seat.

Retrieving Large Volumes of Data

By Andrei Dzianisau
Edited by Ian Abramson

Today's database systems are being asked to store more information than ever before. According to research firm IDC, the size of data — which it calls the "digital universe" — will grow to 2.7 zetabytes in 2012, up 48 percent from 2011's record year. This growth is unprecedented, and we must be able to manage databases that contain significantly more information that needs to be retrieved in less and less time. It is more important than ever to understand the options Oracle provides to users within the database that can truly empower your applications regardless of the size of your data sets.

Hardware appliances such as Oracle's Exadata have helped significantly, but keeping the laws of physics in mind, there is a limit. Whether you use the faster memory, solid state disks or read technology, sometimes even very powerful hardware doesn't help. For instance, it's not possible to read 60 GB of data off from disk in just a few seconds. There are limits to how powerful a storage system can be. As a result, we need to understand how to take advantage of Oracle and the features and facilities available to us. Not everyone can afford to purchase an Exadata machine to solve their data volume issues, so this article investigates some of these capabilities that all Oracle databases can take advantage of.

It is a very common situation that a business asks for information to answer important business questions on daily/hourly basis. It then demands that these questions are answered within seconds (or, at very most, a matter of minutes). It is also very common to expect business intelligence (BI) systems to create reports that require vast amounts of data to support complex reporting needs and be able to use that data quickly and efficiently. The tools today try to optimize how they retrieve data, but, sometimes, with poorly deployed data warehouses, this is not always possible.

What are the things you can do today with a basic Oracle installation? This article will discuss numerous strategies and approaches we at EPAM Systems have seen to be very successful in optimizing how we use and deploy data solutions. The top concepts we will discuss include the following:

- Narrowing down amount of data to access
- Using aggregates to optimize data retrieve and minimize calculations
- Changing the design of your solutions
- Using some of the built-in functions supplied by Oracle
- Combining processes to streamline processing

These are just a few of the approaches one could take to support and ensure that your data warehouse performs well and consistently. Each approach we will discuss may appear basic on the surface, but the value each can provide can significantly impact your overall performance and save your organization from investing in unneeded hardware.

Selecting Only the Data You Need

One of the most basic concepts you need to embrace is to minimize how much information is retrieved from disks. Disks tend to be the slowest part of the I/O equation, and, therefore, you need to ensure that only the blocks you are interested in are being read. Selecting data and reducing the number of full table scans is a lifelong journey for many Oracle professionals.

Filters with low selectivity bring a database to conditions when full table scans of very big fact tables are the most efficient way to access and calculate data. For instance, report with the time dimension filter applied on year level like year = 2011 and no other filters provided for products or departments will cause the cost-based optimizer (CBO) to switch to full-table scan (FTS).

Here is a simple example, following query extracts data for whole year 1998:

```
create bitmap index x1 on times(fiscal_month_number) compute statistics;
create bitmap index x2 on times(fiscal_year) compute statistics;
```

The following code shows statement with very few filters with wide selection criteria defined:

```
SELECT
 t.fiscal_year
, t.fiscal_month_number
, c.channel_desc
, SUM(s.quantity_sold)
, SUM(s.amount_sold)
, count(distinct promo_id)
FROM sales s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
and t.fiscal_year = 1998
group by t.fiscal_year, t.fiscal_month_number
, c.channel_desc;
```

continued on page 24

Below is the execution plan for the statement listed above:

```
_____________________________________________________
| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time | Pstart| Pstop |
_____________________________________________________
| 0 | SELECT STATEMENT | | 30 | 1680 | 512 (4)| 00:00:07 | | |
| 1 | HASH GROUP BY | | 30 | 1680 | 512 (4)| 00:00:07 | | |
| 2 | VIEW | VW_DAG_0 | 68 | 3808 | 512 (4)| 00:00:07 | | |
| 3 | HASH GROUP BY | | 68 | 3604 | 512 (4)| 00:00:07 | | |
| 4 | MERGE JOIN | | 68 | 3604 | 511 (4)| 00:00:07 | | |
| 5 | TABLE ACCESS BY INDEX ROWID | CHANNELS | 5 | 65 | 2 (0)| 00:00:01 | | |
| 6 | INDEX FULL SCAN | CHANNELS_PK | 5 | | 1 (0)| 00:00:01 | | |
|* 7 | SORT JOIN | | 68 | 2720 | 509 (4)| 00:00:07 | | |
| 8 | VIEW | VW_GBC_10 | 68 | 2720 | 508 (4)| 00:00:07 | | |
| 9 | HASH GROUP BY | | 68 | 2584 | 508 (4)| 00:00:07 | | |
|* 10 | HASH JOIN | | 191K| 7107K| 502 (3)| 00:00:07 | | |
| 11 | PART JOIN FILTER CREATE | :BF0000 | 304 | 4560 | 10 (10)| 00:00:01 | | |
|* 12 | VIEW | index$_join$_002 | 304 | 4560 | 10 (10)| 00:00:01 | | |
|* 13 | HASH JOIN | | | | | | | |
|* 14 | HASH JOIN | | | | | | | |
| 15 | BITMAP CONVERSION TO ROWIDS| | 304 | 4560 | 1 (0)| 00:00:01 | | |
|* 16 | BITMAP INDEX SINGLE VALUE | X2 | | | | | | |
| 17 | INDEX FAST FULL SCAN | TIMES_PK | 304 | 4560 | 8 (0)| 00:00:01 | | |
| 18 | BITMAP CONVERSION TO ROWIDS | | 304 | 4560 | 1 (0)| 00:00:01 | | |
| 19 | BITMAP INDEX FULL SCAN | X1 | | | | | | |
| 20 | PARTITION RANGE JOIN-FILTER | | 918K| 20M| 489 (2)| 00:00:06 |:BF0000|:BF0000|
| 21 | TABLE ACCESS FULL | SALES | 918K| 20M| 489 (2)| 00:00:06 |:BF0000|:BF0000|
_____________________________________________________
```

Wide selection criteria specified for query forces CBO to perform a FTS on the SALES table as the most effective access path.

The definition of additional filters that decreases the amount of records to be accessed will change FTS to INDEX access path.

But what if the business user requires information for just one month and information for the rest of the months is not really required?

The following code contains an additional filter, t.fiscal_month_number = 1, that restricts query to access fewer rows in the database

```
SELECT
 t.fiscal_year
, t.fiscal_month_number
, c.channel_desc
, SUM(s.quantity_sold)
, SUM(s.amount_sold)
, count(distinct promo_id)
FROM sales s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
and t.fiscal_year = 1998
and t.fiscal_month_number = 1
group by t.fiscal_year, t.fiscal_month_number
, c.channel_desc;
```

The explain plan changes significantly by this change to the SQL query. In the previous select statement, we read the SALES table with a FTS. By changing the query and adding the additional constraints, we now see that we read SALES table via an Index read. This results in a lower cost and a significantly better performing query.

```
_____________________________________________________
| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time | Pstart| Pstop |
_____________________________________________________
| 0 | SELECT STATEMENT | | 3 | 168 | 182 (2)| 00:00:03 | | |
| 1 | HASH GROUP BY | | 3 | 168 | 182 (2)| 00:00:03 | | |
| 2 | VIEW | VW_DAG_0 | 8 | 448 | 182 (2)| 00:00:03 | | |
| 3 | HASH GROUP BY | | 8 | 408 | 182 (2)| 00:00:03 | | |
|* 4 | HASH JOIN | | 15961 | 794K| 181 (1)| 00:00:03 | | |
| 5 | TABLE ACCESS FULL | CHANNELS | 5 | 65 | 3 (0)| 00:00:01 | | |
| 6 | NESTED LOOPS | | | | | | | |
| 7 | NESTED LOOPS | | 15961 | 592K| 177 (0)| 00:00:03 | | |
| 8 | TABLE ACCESS BY INDEX ROWID | TIMES | 25 | 375 | 7 (0)| 00:00:01 | | |
| 9 | BITMAP CONVERSION TO ROWIDS | | | | | | | |
| 10 | BITMAP AND | | | | | | | |
|* 11 | BITMAP INDEX SINGLE VALUE | X1 | | | | | | |
|* 12 | BITMAP INDEX SINGLE VALUE | X2 | | | | | | |
| 13 | PARTITION RANGE ITERATOR | | | | | KEY | KEY |
| 14 | BITMAP CONVERSION TO ROWIDS | | | | | | | |
|* 15 | BITMAP INDEX SINGLE VALUE | SALES_TIME_BIX | | | | KEY | KEY |
| 16 | TABLE ACCESS BY LOCAL INDEX ROWID| SALES | 629 | 14467 | 177 (0)| 00:00:03 | 1 |
1 |
_____________________________________________________
```

This simple example just says: "Don't wait for a miracle just because you have very smart software and very powerful hardware." Adding some more constraints, you still can answer the questions businesses ask with less time.

This approach is not suitable for every business scenario, and what will happen if people still ask for a whole year or even several years of information?

The next topic will highlight methods that are actually using the same approach: Reduce the amount of records accessed by the system to answer your question.

Data Aggregation and Query Rewrite

In case a business scenario doesn't allow the use of filters to query a small set of records, and access to large volume of rows is required with full tables scans performed, then the use of materialized views with aggregations can be a solution. QUERT REWRITE is a very powerful feature that allows switch query to use different data storage without the necessity to manually rewrite query, especially when the system-generated SQL is used.

Case 1: Simple aggregation with a query rewrite for two levels of time hierarchy

The following example shows how to change a query to access MV with aggregated data instead of actual tables listed in the query code. Consider an application that regularly is required to create a report that can drill down from aggregated by year and also needs to look at data at the quarterly or monthly levels. The report will query the data; the following is the query that would be issued if we wanted to retrieve the data summarized at the year level.

```
SELECT
 t.fiscal_year
, c.channel_desc
, SUM(s.quantity_sold)
, SUM(s.amount_sold)
FROM sales s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
and t.fiscal_year = 1998
group by t.fiscal_year
, c.channel_desc;
```

Below is the explain plan for the query listed above:

```
-------------------------------------------------------------------------------------------------
| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time | Pstart| Pstop |
-------------------------------------------------------------------------------------------------
| 1 | HASH GROUP BY | | 4 | 212 | 520 (4)| 00:00:07 | | |
| 0 | SELECT STATEMENT | | 4 | 212 | 520 (4)| 00:00:07 | | |
| 2 | VIEW | VW_DAG_0 | 8 | 424 | 520 (4)| 00:00:07 | | |
| 3 | HASH GROUP BY | | 8 | 400 | 520 (4)| 00:00:07 | | |
| 4 | MERGE JOIN | | 8 | 400 | 519 (4)| 00:00:07 | | |
| 5 | TABLE ACCESS BY INDEX ROWID | CHANNELS | 5 | 65 | 2 (0)| 00:00:01 | | |
| 6 | INDEX FULL SCAN | CHANNELS_PK | 5 | | 1 (0)| 00:00:01 | | |
|* 7 | SORT JOIN | | 8 | 296 | 517 (4)| 00:00:07 | | |
| 8 | VIEW | VW_GBC_10 | 8 | 296 | 516 (4)| 00:00:07 | | |
| 9 | HASH GROUP BY | | 8 | 280 | 516 (4)| 00:00:07 | | |
|* 10 | HASH JOIN | | 191K| 6546K| 510 (3)| 00:00:07 | | |
| 11 | PART JOIN FILTER CREATE | :BF0000 | 304 | 3648 | 18 (0)| 00:00:01 | | |
|* 12 | TABLE ACCESS FULL | TIMES | 304 | 3648 | 18 (0)| 00:00:01 | | |
| 13 | PARTITION RANGE JOIN-FILTER| | 918K| 20M| 489 (2)| 00:00:06 |:BF0000|:BF0000|
| 14 | TABLE ACCESS FULL | SALES | 918K| 20M| 489 (2)| 00:00:06 |:BF0000|:BF0000|
-------------------------------------------------------------------------------------------------
```

The main problem with this query is the partition range scan on the fact table (or full table scan if this table is not partitioned). Even having the bitmap index built on `TIMES.FISCAL_YEAR` column, the selection criteria is too wide and full table scans seem reasonable.

A materialized view (MV) with aggregation for fact columns can successfully eliminate full table scans on big fact tables and change it to MV access very easily. An additional column `CHANNEL_DESC` is provided here to create an example of data aggregation across various dimensions, while the query above will still use the newly create MV instead of a fact table.

Create a materialized view

The following is a materialized view creation statement that materializes output of the query:

```
create materialized view MV_RW1 build immediate
using index refresh force on demand enable query rewrite
as
SELECT
 t.fiscal_year
, t.fiscal_quarter_desc
, c.channel_desc
, s.promo_id
, SUM(s.quantity_sold)
, SUM(s.amount_sold)
FROM sales s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
group by t.fiscal_year, fiscal_quarter_desc
, c.channel_desc
, s.promo_id;
```

Check if this materialized view can be picked up by the optimizer for rewrite by executing following code:

```
DECLARE
 v VARCHAR2 (32000);
 att SYS.RewriteArrayType;
BEGIN
 v :=
 q'!SELECT
 t.fiscal_year
 , SUM(s.quantity_sold)
 , SUM(s.amount_sold)
FROM sales s
 , times t
 , channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
group by t.fiscal_year
 , c.channel_desc
 !';
 DBMS_MVIEW.explain_rewrite (query => v, mv=>'MV_RW1', statement_id => 'rw1');
 Commit;
END;
/
```

The code listed above requires the additional table `REWRITE_TABLE` to be created in advance. Obtain the script for this table creation from <ORACLE_HOME>/rdbms/admin/utlxrw.sql script.

Information about query rewrite status can be displayed by following this query:

```
SELECT statement_id, mv_name, sequence, message, pass, mv_in_msg FROM REWRITE_TABLE
where statement_id = 'rw1'
order by sequence;
```

The output may look like the following:

```
STATEMENT_ MV_NAME SEQUENCE MESSAGE PASS MV_IN_MSG
---------- ------- -------- ------- ---- ----------
rw1 MV_RW1 1 QSM-01151: query was NO
 rewritten

rw1 MV_RW1 2 QSM-01033: query rew YES MV_RW1
 ritten with material
 ized view, MV_RW1
```

continued on page 26

This output provides information about a successful query rewrite performed by the optimizer. A simple execution plan check confirms that query rewrite occurs:

```
explain plan for
SELECT
 t.fiscal_year
, SUM(s.quantity_sold)
, SUM(s.amount_sold)
, count(distinct promo_id)
FROM sales s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
and t.fiscal_year = 1998
group by t.fiscal_year
, c.channel_desc;

select * from table(dbms_xplan.display);

_____________________________________
| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
_____________________________________
| 0 | SELECT STATEMENT | | 3 | 93 | 4 (25)| 00:00:01 |
| 1 | HASH GROUP BY | | 3 | 93 | 4 (25)| 00:00:01 |
|* 2 | MAT_VIEW REWRITE ACCESS FULL| MV_RW1 | 14 | 434 | 3 (0)| 00:00:01 |
_____________________________________

Predicate Information (identified by operation id):
___________________

2 - filter("MV_RW1"."FISCAL_YEAR"=1998)
```

The execution plan shows that the materialized view we've created is picked up by the optimizer instead of tables listed in the query code.

Case 2: Query equivalence

Sometimes systems have fact metrics, defined as unique numbers of something. In such cases, it is very hard to apply a query rewrite, as report drilldowns will require such metrics to be aggregated exactly to the selected level of hierarchy.

Considering the following query as an example:

One of the fact metrics had been designed as `COUNT(DISTINCT PROMO_ID)`. The appearance of this expression in the query code will invalidate the materialized view MV_RW1 created earlier. The query below illustrates how the query only changes slightly but has a significant impact to the way which Oracle executes the query. This change results in the query not using the Materialized View and now executing as a standard SQL statement as you may see when you look at the explain plan following the query.

```
explain plan for
SELECT
 t.fiscal_year
, c.channel_desc
, SUM(s.quantity_sold)
, SUM(s.amount_sold)
, count(distinct promo_id)
FROM sales s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
and t.fiscal_year = 1998
group by t.fiscal_year
, c.channel_desc;

select * from table(dbms_xplan.display);

_____________________________________________________
| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time | Pstart| Pstop |
_____________________________________________________
| 0 | SELECT STATEMENT | | 4 | 212 | 520 (4)| 00:00:07 | | |
| 1 | HASH GROUP BY | | 4 | 212 | 520 (4)| 00:00:07 | | |
| 2 | VIEW | VW_DAG_0 | 8 | 424 | 520 (4)| 00:00:07 | | |
| 3 | HASH GROUP BY | | 8 | 400 | 520 (4)| 00:00:07 | | |
| 4 | MERGE JOIN | | 8 | 400 | 519 (4)| 00:00:07 | | |
| 5 | TABLE ACCESS BY INDEX ROWID | CHANNELS | 5 | 65 | 2 (0)| 00:00:01 | | |
| 6 | INDEX FULL SCAN | CHANNELS_PK | 5 | | 1 (0)| 00:00:01 | | |
|* 7 | SORT JOIN | | 8 | 296 | 517 (4)| 00:00:07 | | |
| 8 | VIEW | VW_GBC_10 | 8 | 296 | 516 (4)| 00:00:07 | | |
| 9 | HASH GROUP BY | | 8 | 280 | 516 (4)| 00:00:07 | | |
|* 10 | HASH JOIN | | 191K| 6546K| 510 (3)| 00:00:07 | | |
| 11 | PART JOIN FILTER CREATE | :BF0000 | 304 | 3648 | 18 (0)| 00:00:01 | | |
|* 12 | TABLE ACCESS FULL | TIMES | 304 | 3648 | 18 (0)| 00:00:01 | | |
| 13 | PARTITION RANGE JOIN-FILTER| | 918K| 20M| 489 (2)| 00:00:06 |:BF0000|:BF0000|
| 14 | TABLE ACCESS FULL | SALES | 918K| 20M| 489 (2)| 00:00:06 |:BF0000|:BF0000|
_____________________________________________________
```

The fact that the query changes the execution plan means that we need to improve our previous definition for the materialized view. The `COUNT(DISTINCT)` metric allows query rewrite for all business scenarios in only one case, so by moving the field used in the expression into a list of fields used in selection criteria and grouping by part of the query we can improve performance and maximize query rewrite.

The materialized view creation code with the `S.PROMO_ID` column added into the list of fields of materialized view is below:

```
create materialized view MV_RW2 build immediate using index refresh force on demand enable
query rewrite
as
SELECT
 t.fiscal_year
, t.fiscal_quarter_desc
, c.channel_desc
, s.promo_id
, SUM(s.quantity_sold)
, SUM(s.amount_sold)
FROM sales s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
group by t.fiscal_year, fiscal_quarter_desc
, c.channel_desc
, s.promo_id
;
```

The explain plan for a query that contains the COUNT(DISTINCT PROMO_ID) metric is below:

```
explain plan for
SELECT
 t.fiscal_year
, c.channel_desc
, SUM(s.quantity_sold)
, SUM(s.amount_sold)
, count(distinct promo_id)
FROM sales s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
and t.fiscal_year = 1998
group by t.fiscal_year
, c.channel_desc;

select * from table(dbms_xplan.display);

-----------------------------------------------------------------------------
| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
-----------------------------------------------------------------------------
| 0 | SELECT STATEMENT | | 3 | 93 | 4 (25)| 00:00:01 |
| 1 | SORT GROUP BY | | 3 | 93 | 4 (25)| 00:00:01 |
|* 2 | MAT_VIEW REWRITE ACCESS FULL| MV_RW2 | 14 | 434 | 3 (0)| 00:00:01 |
-----------------------------------------------------------------------------
```

The generated output shows that the query was rewritten by CBO.

This solution can be adopted if the amount of records in the MV is much less than in the fact table. However, in case the total row count of MV is close to the table, it doesn't make any sense to keep this MV in the database.

A quick and nasty solution to the problem is a query equivalence feature used along with the multilevel aggregation used for MV creation:

```
create materialized view MV_RW3 build immediate using index refresh force on demand enable
query rewrite
as
SELECT
 t.fiscal_year
, t.fiscal_quarter_desc
, c.channel_desc
, SUM(s.quantity_sold)
, SUM(s.amount_sold)
, count(distinct promo_id)
, grouping(t.fiscal_year) grpyear
, grouping (t.fiscal_quarter_desc) grpquarter
, grouping(c.channel_desc) grpchannel
FROM sales s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
Group by cube(fiscal_year, fiscal_quarter_desc, channel_desc)
```

This statement will produce several levels of aggregation with dedicated information about totals and subtotals produced by query. The columns GRPYEAR, GRPCHANNEL and GRPQUARTER, defined using the GROUPING function, will generate information about multiple levels of aggregates generated by query.

```
declare

v_sql1 varchar2(6000);
v_sql2 varchar2(600);

begin

v_sql1 := q'!
SELECT
 t.fiscal_year
, c.channel_desc
, SUM(s.quantity_sold)
, SUM(s.amount_sold)
, count(distinct promo_id)
FROM sales s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
group by t.fiscal_year
, c.channel_desc
!';

v_sql2 := q'!
select fiscal_year, channel_desc, quantity_sold, amount_sold, cnt_promo_id
from MV_RW3
where grpyear = 0
and grpchannel = 0
and grpquarter = 1
!';

sys.DBMS_ADVANCED_REWRITE.DECLARE_REWRITE_EQUIVALENCE ( 'TEST', v_sql1, v_sql2,
FALSE, 'GENERAL');

end;
/
```

It is necessary to set the QUERY_REWTITE_INTEGRITY parameter to the TRUSTED value to make prebuilt materialized views picked up by CBO.

```
ALTER SESSION SET QUERY_REWRITE_INTEGRITY = TRUSTED;
```

By checking the execution plan for the statement, we will see that our query equivalence is used by CBO and the materialized view appears in the plan:

continued on page 28

```
explain plan for
SELECT
 t.fiscal_year
, c.channel_desc
, SUM(s.quantity_sold)
, SUM(s.amount_sold)
, count(distinct promo_id)
FROM sales s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
and t.fiscal_year = 1998
group by t.fiscal_year
, c.channel_desc;

select * from table(dbms_xplan.display);

_____
| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
_____
| 0 | SELECT STATEMENT | | 2 | 72 | 3 (0)| 00:00:01 |
|* 1 | MAT_VIEW ACCESS FULL| MV_RW3 | 2 | 72 | 3 (0)| 00:00:01 |
_____

Predicate Information (identified by operation id):
_____
1 - filter("FISCAL_YEAR"=1998 AND "GRPYEAR"=0 AND "GRPCHANNEL"=0 AND
"GRPQUARTER"=1)
```

It is possible to create many different query equivalences for different scenarios, and all these equivalences will be used instead of the original queries.

Most Recent Features: Recursive Subquery Factoring

Hierarchical queries have always been a big pain for databases. Oracle's CONNECT BY statement provided the ability to extract data hierarchies for a very long time with no analogues in DBMS of other vendors. With Oracle Database 11*g*R2, we were provided with a new feature that follows the SQL 99 standard of recursive common table expressions (also implemented by MSSQL, DB2 and Postgre SQL) that I expect is being underutilized but can provide you significant value. This is an extension of SQL that allows recursive/hierarchical queries along with CONNECT BY statements. If you recall, Oracle's CONNECT BY command is a great feature with one exception: It is really difficult to make it work quickly. This command looks much like the original CONNECT BY command with some small, but impactful, differences that provide us with the improvements we need.

Let's look at how things change with the new recursive subquery versus the old CONNECT BY clause. First, here is the original select statement using a CONNECT BY clause:

```
SELECT ROWNUM ROW_NUM,
LEVEL LEVEL_NUM,
FLEX_VALUE_SET_ID,
PARENT_FLEX_VALUE,
RANGE_ATTRIBUTE,
CHILD_FLEX_VALUE_LOW,
CHILD_FLEX_VALUE_HIGH,
LAST_UPDATE_DATE,
LAST_UPDATED_BY,
CREATION_DATE,
CREATED_BY,
LAST_UPDATE_LOGIN,
START_DATE_ACTIVE,
END_DATE_ACTIVE
 FROM FND_FLEX_VALUE_NORM_HIERARCHY
START WITH RANGE_ATTRIBUTE = 'C'
CONNECT BY PRIOR FLEX_VALUE_SET_ID = FLEX_VALUE_SET_ID
 AND PRIOR PARENT_FLEX_VALUE BETWEEN CHILD_FLEX_VALUE_LOW AND CHILD_FLEX_
VALUE_HIGH
 AND RANGE_ATTRIBUTE = 'P'
 AND PRIOR PARENT_FLEX_VALUE <> PARENT_FLEX_VALUE;
```

Below are query execution statistics:

```
2173299 rows selected.

Elapsed: 27:55:08.36

Execution Plan
_____
Plan hash value: 3821678197

_____
| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
_____
| 0 | SELECT STATEMENT | | 430K| 150M| 2787 (84)| 00:00:51 |
| 1 | COUNT | | | | | |
|* 2 | CONNECT BY NO FILTERING WITH START-WITH| | | | |
| 3 | TABLE ACCESS FULL | FND_FLEX_VALUE_NORM_HIERARCHY | 430K| 22M| 463 (4)|
00:00:09 |
_____

Predicate Information (identified by operation id):
_____

 2 - access("FLEX_VALUE_SET_ID"=PRIOR "FLEX_VALUE_SET_ID")
 filter("CHILD_FLEX_VALUE_LOW"<=PRIOR "PARENT_FLEX_VALUE" AND "CHILD_FLEX_
VALUE_HIGH">=PRIOR
 "PARENT_FLEX_VALUE" AND "RANGE_ATTRIBUTE"='P' AND "PARENT_FLEX_VALUE"<>PRIOR
"PARENT_FLEX_VALUE" AND
 "RANGE_ATTRIBUTE"='C')


Statistics
_____
 716 recursive calls
 2 db block gets
 1017 consistent gets
 874 physical reads
 0 redo size
 132911624 bytes sent via SQL*Net to client
 1594222 bytes received via SQL*Net from client
 144888 SQL*Net roundtrips to/from client
 13 sorts (memory)
 0 sorts (disk)
 2173299 rows processed
```

The query was running for 27 hours on 400 K records table and producing approximately 2 M records output, which is really slow.

The rewrite of the CONNECT BY query using the recursive subquery factoring feature appeared in Oracle Database 11*g*R2 displayed outstanding performance comparing to original version of the query.

Below is the recursive subquery code:

```
WITH hier (LEVEL_NUM, FLEX_VALUE_SET_ID, PARENT_FLEX_VALUE, RANGE_ATTRIBUTE,
CHILD_FLEX_VALUE_LOW, CHILD_FLEX_VALUE_HIGH, LAST_UPDATE_DATE,
LAST_UPDATED_BY, CREATION_DATE, CREATED_BY, LAST_UPDATE_LOGIN,
START_DATE_ACTIVE, END_DATE_ACTIVE)
 AS (SELECT 1 LEVEL_NUM,
 FLEX_VALUE_SET_ID,
 PARENT_FLEX_VALUE,
 RANGE_ATTRIBUTE,
 CHILD_FLEX_VALUE_LOW,
 CHILD_FLEX_VALUE_HIGH,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 CREATION_DATE,
 CREATED_BY,
 LAST_UPDATE_LOGIN,
 START_DATE_ACTIVE,
 END_DATE_ACTIVE
 FROM FND_FLEX_VALUE_NORM_HIERARCHY
 WHERE RANGE_ATTRIBUTE = 'C'
 UNION ALL
 SELECT LEVEL_NUM + 1 LEVEL_NUM,
 h.FLEX_VALUE_SET_ID,
 h.PARENT_FLEX_VALUE,
 h.RANGE_ATTRIBUTE,
 h.CHILD_FLEX_VALUE_LOW,
 h.CHILD_FLEX_VALUE_HIGH,
 h.LAST_UPDATE_DATE,
 h.LAST_UPDATED_BY,
 h.CREATION_DATE,
 h.CREATED_BY,
 h.LAST_UPDATE_LOGIN,
 h.START_DATE_ACTIVE,
 h.END_DATE_ACTIVE
 FROM FND_FLEX_VALUE_NORM_HIERARCHY h, hier
 WHERE hier.FLEX_VALUE_SET_ID = h.FLEX_VALUE_SET_ID
 AND hier.PARENT_FLEX_VALUE BETWEEN h.CHILD_FLEX_VALUE_LOW
 AND h.CHILD_FLEX_VALUE_HIGH
 AND h.RANGE_ATTRIBUTE = 'P'
 AND hier.PARENT_FLEX_VALUE <> h.PARENT_FLEX_VALUE)
 SEARCH BREADTH FIRST BY FLEX_VALUE_SET_ID SET dpth
SELECT ROWNUM ROW_NUM,
 LEVEL LEVEL_NUM,
 FLEX_VALUE_SET_ID,
 PARENT_FLEX_VALUE,
 RANGE_ATTRIBUTE,
 CHILD_FLEX_VALUE_LOW,
 CHILD_FLEX_VALUE_HIGH,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 CREATION_DATE,
 CREATED_BY,
 LAST_UPDATE_LOGIN,
 START_DATE_ACTIVE,
 END_DATE_ACTIVE
 FROM hier;
```

Below are the statistics for the reqursive subquery factoring version:

```
2173299 rows selected.

Elapsed: 02:02:28.34

Execution Plan
----------
Plan hash value: 2052308742

-------------------------------------------------------------------------------
| Id | Operation | Name | Rows | Bytes |TempSpc| Cost (%CPU)| Time |
-------------------------------------------------------------------------------
| 0 | SELECT STATEMENT | | 456K| 170M| | 6000 (55)| 00:01:49 |
| 1 | VIEW | | 456K| 170M| | 6000 (55)| 00:01:49 |
| 2 | UNION ALL (RECURSIVE WITH) BREADTH FIRST| | | | | |
|* 3 | TABLE ACCESS FULL | FND_FLEX_VALUE_NORM_HIERARCHY | 215K| 11M| | 462 (3)|
00:00:09 |
|* 4 | HASH JOIN | | 240K| 39M| 13M| 3079 (27)| 00:00:56 |
|* 5 | TABLE ACCESS FULL | FND_FLEX_VALUE_NORM_HIERARCHY | 215K| 11M| | 462 (3)|
00:00:09 |
| 6 | RECURSIVE WITH PUMP | | | | | |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

 3 - filter("RANGE_ATTRIBUTE"='C')
 4 - access("HIER"."FLEX_VALUE_SET_ID"="H"."FLEX_VALUE_SET_ID")
 filter("HIER"."PARENT_FLEX_VALUE">="H"."CHILD_FLEX_VALUE_LOW" AND
 "HIER"."PARENT_FLEX_VALUE"<="H"."CHILD_FLEX_VALUE_HIGH" AND "HIER"."PARENT_
FLEX_VALUE"<>"H"."PARENT_FLEX_VALUE")
 5 - filter("H"."RANGE_ATTRIBUTE"='P')


Statistics
----------
 31 recursive calls
 11350452 db block gets
 12036 consistent gets
 5522 physical reads
 0 redo size
 92347039 bytes sent via SQL*Net to client
 1594222 bytes received via SQL*Net from client
 144888 SQL*Net roundtrips to/from client
 14 sorts (memory)
 0 sorts (disk)
 2173299 rows processed
```

This version of the query was running for just two hours instead of 27 hours, as it was detected for the CONNECT BY query.

Combine Multiple Processes into a Single One

Working on some projects in the past, we have received a requirement for a data summarization task to provide the ability to summarize data by columns defined by users at runtime and also store information about records summarized together. This functionality was required for data audit purposes.

The amount of records for summarization varied from 10 to several millions, so it was very tricky to make the whole process very fast.

continued on page 30

Finally, after some time, we came up with a solution that both summarizes the data and writes information about rows grouped together. Here is the example:

At first step, I will create a copy of the SALES table with a new ID column created as a primary key because a single column PK is required for this solution:

```
create table sales2 as
select rownum id, s1.* from sales s1;
```

As a next step, I will modify the query used for MV creation in the query equivalence example by changing the SALES table to the SALES2 one created by a prior command:

```
SELECT
 s.id
,t.fiscal_year
, t.fiscal_quarter_desc
, c.channel_desc
, SUM(s.quantity_sold) sum_qty_sold
, SUM(s.amount_sold) sum_amt_sold
, grouping(t.fiscal_year) grpyear
, grouping (t.fiscal_quarter_desc) grpquarter
, grouping(c.channel_desc) grpchannel
FROM
 sales2 s
, times t
, channels c
WHERE
 s.time_id = t.time_id
 AND s.channel_id = c.channel_id
 AND fiscal_year = 1998
 AND fiscal_quarter_desc = '1998-01'
GROUP BY
 cube(s.id, fiscal_year, fiscal_quarter_desc, channel_desc)
HAVING
 grouping(t.fiscal_year) =0
 AND grouping (t.fiscal_quarter_desc) =0
 AND grouping(c.channel_desc) =0
ORDER BY
 2,3,4,1 DESC ;
```

This query produces the following output:

```
ID FISCAL_YEAR FISCAL_QUARTER_DESC CHANNEL_DESC SUM_QTY_SOLD SUM_AMT_SOLD GRPYEAR GRPQUARTER GRPCHANNEL
— ———— ———————— ———————— ———————— ———————— ——— ———— ———
<NULL> 1998 1998-01 Direct Sales 31870 4456562.08 0 0 0
43324 1998 1998-01 Direct Sales 1 29.8 0 0 0
43323 1998 1998-01 Direct Sales 1 29.8 0 0 0
…
<NULL> 1998 1998-01 Internet 3754 676867.34 0 0 0
42877 1998 1998-01 Internet 1 94.08 0 0 0
42876 1998 1998-01 Internet 1 94.08 0 0 0
…
<NULL> 1998 1998-01 Partners 6260 1192938.26 0 0 0
43299 1998 1998-01 Partners 1 28.76 0 0 0
43298 1998 1998-01 Partners 1 28.76 0 0 0
…
```

The GROUP BY CUBE clause with the primary key column specified makes the statement generate both a single record from the SALES2 table with joins to the TIMES and CHANNELS tables and totals calculated by the CUBE clause on different combinations of fields specified in the GROUP BY CUBE.

The HAVING clause specified in the query allows us to filter out all subtotals generated for combinations of `fiscal_year`, `fiscal_quarter_desc` and `channel_desc` fields.

As a result, the remaining records are:

- the records from the SALES2 table that fall into query selection criteria; and
- the totals calculated for the group of `fiscal_year`, `fiscal_quarter_desc` and `channel_desc` fields.

These totals are highlighted by yellow.

A similar result set can be obtained by two separate queries:

```
SELECT null as id
, t.fiscal_year
, t.fiscal_quarter_desc
, c.channel_desc
, SUM(s.quantity_sold)
, SUM(s.amount_sold)
FROM sales2 s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
and fiscal_year = 1998
and fiscal_quarter_desc = '1998-01'
Group by (fiscal_year, fiscal_quarter_desc, channel_desc)
;
```

And:

```
SELECT id
, t.fiscal_year
, t.fiscal_quarter_desc
, c.channel_desc
, s.quantity_sold
, s.amount_sold
FROM sales2 s
, times t
, channels c
WHERE s.time_id = t.time_id
AND s.channel_id = c.channel_id
and fiscal_year = 1998
and fiscal_quarter_desc = '1998-01'
```

We need to track both summarized results and exact records that had been summarized together to produce the summary row, so the queries listed above will not allow us to define which records are combined together.

To make this example clearer, I will create two separate tables. The first one is for summarized data, and the second is for records summarized together.

Below is the table for keeping summarized data:

```
create table sum_sales (
 fiscal_year number(4),
 fiscal_quarter_desc varchar2(7),
 channel_desc varchar2(20),
 sum_qty_sold number,
 sum_amt_sold number,
 sum_grp number
);
```

Below is the table for keeping information about rows summarized together:

```
create table det_sales (
 id number,
 sum_grp number
);
```

Both tables contain a SUM_GRP column that will hold the summarization group information for generated totals and rows summarized together.

We will also require a sequence that will generate values for the summarization groups and two functions that will provide access to sequence.nextval and sequence.currval attributes:

Below is the sequence creation code:

```
create sequence seq1 start with 1 increment by 1;
```

Below shows the function returning sequence next value code:

```
create or replace function seq_nextval return number is
 retval number;
begin
 retval := seq1.nextval;
 return retval;
end;
/
```

Below shows the function returning sequence current value code:

```
create or replace function seq_currval return number is
 retval number;
begin
 retval := seq1.currval;
 return retval;
end;
/
```

Combining all SELECT statements together, we will create a statement that assembles both queries into single one using GROUP BY CUBE clause. The MULTI-TABLE INSERT feature will allow us to load data into separate and different tables using a single SQL command. By applying special conditions we can determine which records should go to summary table, and which one — to detail table. In our example below, the data goes to the summary table ID is not null and into the detail table when it is. As a result, SQL statement provided below, combines all required functionality within one statement instead of several ones and ultimately improves performance of your data loads.

```
INSERT
 WHEN ID IS NULL THEN INTO sum_sales
 (fiscal_year, fiscal_quarter_desc, channel_desc,sum_qty_sold, sum_amt_sold, sum_grp)
 VALUES (fiscal_year, fiscal_quarter_desc, channel_desc,sum_qty_sold, sum_amt_sold, seq_
nextval)
 WHEN ID IS NOT NULL THEN INTO det_sales (id, sum_grp)
 VALUES (id, seq_currval)
SELECT
 s.id
,t.fiscal_year
, t.fiscal_quarter_desc
, c.channel_desc
, SUM(s.quantity_sold) sum_qty_sold
, SUM(s.amount_sold) sum_amt_sold
, grouping(t.fiscal_year) grpyear
, grouping (t.fiscal_quarter_desc) grpquarter
, grouping(c.channel_desc) grpchannel
FROM
 sales2 s
, times t
, channels c
WHERE
 s.time_id = t.time_id
 AND s.channel_id = c.channel_id
 AND fiscal_year = 1998
 AND fiscal_quarter_desc = '1998-01'
– AND channel_desc = 'Partners' –Partners
GROUP BY
 cube(s.id, fiscal_year, fiscal_quarter_desc, channel_desc)
HAVING
 grouping(t.fiscal_year) =0
 AND grouping (t.fiscal_quarter_desc) =0
 AND grouping(c.channel_desc) =0
ORDER BY
 2,3,4,1 DESC ;
```

There are several points I want to emphasize on this statement:

- The ORDER BY construct will order the result set in a way where totals are immediately followed by detailed records that had been used to produce the summary row.
- The functions SEQ_NEXTVAL and SEQ_CURRVAL, which we created before the multitable insert, are used in the list of values of the INSERT, they will generate values for groups of records and will connect totals generated by statement with detailed records.
- The usage of functions for obtaining sequence values is necessary here as the sequence doesn't generate next value for each row of result set in this case. If we will use direct sequence calls in the INSERT part of the statement, then the call to sequence.nextval will be performed for each row no matter if the condition of this operations is equal to TRUE or FALSE, resulting in slower performance.

continued on page 32

After executing this statement, we will have two tables — SUM_SALES and DET_SALES — populated with data, as shown below:

```
select * from sum_sales;

FISCAL_YEAR FISCAL_QUARTER_DESC CHANNEL_DESC SUM_QTY_SOLD SUM_AMT_SOLD
SUM_GRP
———— ———————— ———————— ———— ———— ———
 1998 1998-01 Direct Sales 31870 4456562.08 1
 1998 1998-01 Internet 3754 676867.34 2
 1998 1998-01 Partners 6260 1192938.26 3

select * from det_sales;

ID SUM_GRP
— ———
40234 1
40233 1
40232 1
...
```

The SUM_SALES table contains totals calculated for fiscal year, quarter and channel.

The DET_SALES table contains values of the SALES2.ID column for rows that have been summarized together. The column SUM_GRP in both tables defines the group of summarization. By joining these two tables on SUM_GRP table, you can determine which records from the SALES2 table have been summarized to produce certain row of the SUM_SALES table.

About the Author

Andrei Dzianisau is a software engineering manager at EPAM Systems, a global solutions provider delivering to customers around the globe. Dzianisau's experience working with Oracle Database began more than 15 years ago, beginning with the release of Oracle 7.3. He joined EPAM in 2000 where he's worked with numerous clients to build and maintain data warehouse systems with huge volume databases. His strong experience in SQL, PL/SQL scripting, ETL design and performance tuning of databases helps him get the most out of the Oracle Database. Most recently, he has participated in projects that include Oracle BI applications and Oracle BI performance optimization. Dzianisau lives and works in Minsk, Belarus.

A Multilayered Approach to Oracle Database Availability

By Tom Sager
Edited by Arup Nanda

Not too many years ago, there were few choices to be made when it came to deploying critical Oracle Database servers. Physical servers were purchased, usually high-end RISC UNIX boxes, and some manner of clustering and/or log shipping was implemented to provide for availability. Today, there are many more choices available to an IT organization when it comes to Oracle infrastructure, mostly due to the rise of Intel architecture in this space and its ability to provide the level of performance that was once strictly the domain of high-end servers. This new world of choices has also opened the door to new ideas and new ways of thinking about old problems. One such new idea is a reconsideration of high-availability (HA) and disaster recovery (DR).

HA and DR the Oracle Way

RAC has been the first choice for high-availability for years now, and it is high-quality software that does an exceptional job. However, it comes at a high price, including:

- Licensing costs
- Infrastructure needs (storage and networking far beyond what you can get by with on a standalone non-clustered database server)
- Labor (far above and beyond what it takes to support non-RAC databases)

Data Guard is most often considered as a solution for DR rather than HA. However, it does provide for automatic Fast-Start Failover via the observer role feature. The issue with implementing this is the need for a third location for the observer, in addition to robust, reliable networking among all three nodes.

Oracle's recommendation for a combined HA/DR environment is RAC in two different data centers and Data Guard between them (refer to their MAA white papers). This is certainly a workable approach, but it's also an expensive one.

Another approach is to configure four physical servers (two in each location), with one primary and three standbys. One of the standbys (in the opposite location from the primary) can also have the observer installed. In this manner, HA is implemented in the local data center via Data Guard Fast-Start Failover while DR redundancy is maintained via additional standby servers (two in this example, because HA redundancy will be needed at the DR site should it ever become primary) in the alternate data center. Most would consider this approach overly complicated and somewhat wasteful of resources, but it does save the expense of RAC licenses.

The Rise of Database Server Virtualization

It has always been a struggle to find the "sweet spot" between cost, complexity and functionality for Oracle Database HA and DR. Adding to the cost component is the fact that the number of cores per processor has been steadily increasing (especially Intel architectures), from two, to four, and now to six and eight. Since Oracle is licensed by core, there are significant ramifications to license costs as databases make their way to these new multicore architectures.

The Intel multicore growth is commonly a deciding factor when organizations start making the move to virtual servers. When this move is made, Oracle licensing is "moved up" to the physical host level, allowing for the provisioning of as many virtual Oracle Database servers as the hosts can support. If managed carefully, this approach is a very effective way to get maximum utilization out of the physical servers.

Once this migration is under way, some important added benefits beyond license management become evident:

- Automatic hardware HA
- Virtual switch networking
- Load balancing
- Very fast server reboots
- Dynamic resource provisioning (CPU, memory, disks can all be added "on-the-fly")
- Very quick and easy server deployment (server templates)

This paper is not about the merits of virtualization for Oracle Database servers, so details on the above points will be omitted. They are mentioned here because they have a bearing on how and why new approaches to Oracle HA and DR might be considered. Subject matter experts should be consulted to discuss and verify these aspects of the specific virtualization technology implemented in an organization.

For the purposes of this paper, VMware© server virtualization is assumed.

New Storage Options

Most organizations have been using storage area networks (SAN) for their database storage for years now. One feature that most SANs offer is storage replication. This is often a hardware-level feature, licensed by amount of data being replicated. So although it is neither a new option nor a free one, its deployment against virtual servers opens up new possibilities.

In the VMware world, a virtual machine (VM) server is completely contained in a physical "datastore" (there are other options, but this is the default). Thus, the entire server (OS, DB kernel, DB, logs, etc.) is contained in a small collection of files in a single folder. Each folder represents a distinct server and is itself contained in a datastore (which can loosely be thought of as the VMware counterpart to the SAN LUN).

continued on page 34

The implication of this approach is that selected servers can be located on replicated datastores that keep a redundant copy updated in another location, such as an alternate data center. In the event of a loss at the primary data center, the replicated LUNs can be activated at the alternate data center and their associated datastores added to the inventory of available VM host servers at that location.

The Multilayered HA/DR Approach

The above pieces can fit together to provide a multilayered solution for providing robust HA/DR. Although not without costs, the infrastructure needed is often already available in many IT organizations because both virtualization and SAN replication is becoming ubiquitous. That being the case, taking advantage of these features is an inexpensive way to approach Oracle Database HA and DR together.

As previously mentioned, VMware virtualization is assumed. Other implementations of virtualized servers may provide these functions in slightly different ways, so investigation and testing will be necessary to verify the equivalent functionality and benefits.

No specific SAN technology is assumed, because most of the enterprise-level SANs already provide this type of functionality.

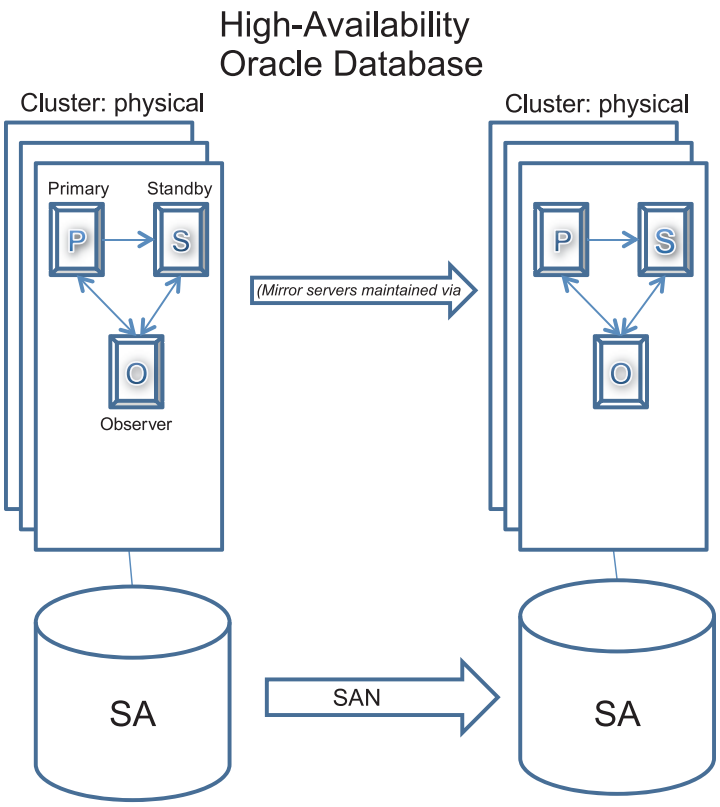


Figure 1: Mirrored Virtual Servers

The approach is simply this:

- One primary DB server, one standby DB server and one observer server
- All three servers are VMs
- All three VM servers run on the same physical host
- Physical VM hosts are clustered (at least three nodes)
- SAN datastore replicated to SAN in alternate data center (ensure write order consistency is enabled)

Availability at each level is provided by a specific technology:

- Database Instance HA: Oracle DataGuard Fast-Start-Failover
- Operating System HA: Oracle DataGuard Fast-Start-Failover
- Hardware HA: Virtual Server Cluster HA (automatic node failover)
- DR: SAN replication

There are of course pros and cons, but in general the pros outweigh the cons in all cases except for the largest databases.

Pros:

- Lots of redundancy: four copies of data, at least six physical servers (assuming minimum three-node virtual host clusters)
- Automatic failover on physical server failure: VMware provides for automatic failover of VMs on a physical host that has failed. The VMs on the failed host are dynamically moved to surviving nodes and started automatically.
- Automatic failover on virtual server failure: If the primary database VM suffers an OS kernel crash or some other issue that makes the server unavailable, Data Guard's Fast-Start Failover will promote the standby database to primary status. Moreover, VMware will recognize some types of server outages and automatically restart the VMs in those circumstances.
- Automatic failover on database failure: Data Guard Fast-Start Failover handles the same way as a server failure.
- No DBA involvement needed for DR: The VM and SAN administrators do their thing to activate/register the virtual servers in the alternate data center.
- Good performance and easily adjusted with dynamically configurable processor and memory provisions at the virtual level: In VMware, both the CPU and RAM can be increased while the server is running.
- No "false positives" in Data Guard due to network glitches: Data Guard is very good at recovering from brief network glitches, but the associated errors in the alert logs are an annoyance. Using Data Guard over a "virtual switch" means no real networking at all; the communication between primary, standby and observer all happens in memory (assuming all three servers on the same VLAN and same switch). This in-memory communication between primary, standby and observer also means the FastStartFailoverThreshold can be set very low (six seconds is the minimum allowed setting). This means that failover will occur quickly in the event the primary server or database fails, which, in turn, means the users' availability experience is much improved as a lengthy hang has been avoided.
- Rolling OS patches are possible without perceived downtime: At most, a brief interruption is caused by switching the primary role to the standby server one time. Many of today's applications tolerate this well, and the users are unaffected. Primary and standby servers are identically configured, so there is no reason to "switch back" after patching.
- Virtual server reboots are lightning fast compared to a physical server.
- Spreads both the risks and responsibilities: No single group or technology is completely responsible for database availability.

- Allows for individual groups to concentrate on core skills: DBAs can focus on the database and let the virtualization team focus on that technology as well as the storage team to focus on the SAN. The DBAs no longer have to be part cluster administrator and part storage administrator.
- Assuming all the virtual server clusters are licensed for Oracle already, this solution is very low cost: The only significant cost is the SAN replication (see below).

Cons:

- SAN replication can be expensive: For example, it may be licensed by the TB, so this approach will add to the cost.
- Four copies of the database are taking up SAN storage: primary plus standby times two due to SAN replication.
- A spanned subnet is needed that covers both primary and standby server locations: This allows the IP address to remain the same regardless of which data center the DB servers are active on.

This approach will obviously not always be the best way to provide for HA/DR in an Oracle Database environment. However, it is very often "good enough" for the majority of application databases in an organization, especially so if enterprise-level virtualization and storage area networks are already in use. The specific technical details of how these technologies work are less important than the fact that they do work already for non-database servers.

It is recognized that some DBAs will not perceive this form of HA/DR in a positive light, because two of the most critical components of traditional database administration (providing high availability and disaster recovery) are in large part being ceded to other groups in the IT organization. Fortunately, in most companies, the DBA is in the best position to make a judgment about the suitability of this approach because they have firsthand knowledge of both the quality of their infrastructure providers (servers, storage and networking) and of their clients' business needs.

Conclusion

Results can be very good with this approach. Much of the success seen with this type of implementation can be attributed to the IT organization's virtualization and storage teams and the fact that they are not doing anything differently for the database tier than they do for other applications. This has always been a problem with RAC: It is an outlier. It places demands on the server, storage and network groups that are not necessarily part of their routine services. With this approach, the only services needed are those already being provided. Much like Intel servers were adopted because of their commodity status, this approach to HA/DR can be adopted based on the commoditization of the infrastructure services available within the organization.

About the Author

Tom Sager is the team leader of the database administration team at Louisville Gas & Electric and Kentucky Utilities. He has more than 20 years of experience as a DBA, focusing primarily on Oracle and SQL Server. Sager has been an IOUG member for 15 years and has authored more than 50 articles on Oracle, SQL Server, Windows and DB2. His area of focus is on virtualization, consolidation, availability, performance and finding new ways to provide for all four simultaneously. Sager may be contacted at **tom.sager@gmail.com**.

Tip From the IOUG SELECT Editors

Stay Up to Date on Security Fixes with Oracle's Security Patch Updates

Oracle is in the process of changing the name for what were previously known as Critical Patch Updates (CPU) to Security Patch Updates (SPU). Both represent the regularly scheduled group of security fixes for Oracle products. Regardless of what Oracle calls them, it is important for DBAs to both investigate as well as implement the recommendations. Security patching is becoming critical no matter your current level of protection. Vulnerabilities can occur at different levels of the software stack as well as between different Oracle components.

Technology is changing at a more rapid pace than ever; you need to be kept informed when Oracle releases critical information related to security. Keep yourself proactively informed by signing up to receive emails alerts. Go to the Oracle Technology Network (OTN) website, **http://otn.oracle.com**, and click on *Critical Patch Updates* on the Essential Links sidebar. (The website has not changed to the new name yet.) Clicking that link brings you to the Critical Patch Updates, Security Alerts and Third Party Bulletins page.

There you will be able to configure email notifications. This only requires an OTN account (which is free; My Oracle Support access is not required). Older alerts are still available on the same OTN page.

Look for more security-related information in the next *SELECT Journal*, Q2 2013.

ORACLE®

ACE PROGRAM

The Oracle ACE program recognizes excellence in the Oracle technology and applications communities, and rewards individuals who generously share their technical knowledge and experiences. Learn more about the **Oracle ACE Program**.

Interview with an Oracle ACE: Gokhan Atil

IOUG: When did you become an Oracle ACE?

Atil: I became an Oracle ACE in September 2011.

IOUG: What is your current occupation?

Atil: I'm a senior database administrator, working in one of the leading IT consultancy companies in Turkey.

IOUG: Tell us a little about your history with Oracle.

Atil: I have been working with Oracle technologies for more than 10 years. I started my career as a software developer, worked with different programming languages and technologies such as Delphi, Visual Basic, PHP and Oracle Forms. After spending a few years with developing software, I realized that I like to work with database technologies, and then I focused on Oracle databases. I feel lucky because I really love my job. So I spend my time on studying, reading about Oracle technologies, sharing my knowledge on my blog, and speaking at local conferences. All these activities make me happy and helped me to receive the Oracle ACE award in 2011.

IOUG: What does this experience mean for you personally and professionally?

Atil: I have a blog where I share my knowledge and experience related to Oracle technologies. I have also been attending Oracle events as a speaker and am helping Oracle users via email by replying to their questions. I wouldn't call myself an Oracle expert or a community leader, but I enjoy researching, writing and sharing Oracle technologies. The Oracle ACE award was very important in order to show that my efforts are recognized by Oracle. I didn't plan to be an Oracle ACE, and didn't see it as a career goal, but it's surely a milestone in my career.

IOUG: Has your status as an Oracle ACE helped you in your career?

Atil: Oracle ACE is definitely a respected title. After becoming an Oracle ACE, more people started to recognize me in the Oracle community. I think it's early to evaluate how it affected to my career, though, because it has been only a one-year adventure for me.

IOUG: What Oracle technology/application are you most looking forward to?

Atil: I'm eager to test the new features of Oracle Database 12*c*. It seems it will come with very handy features such as adaptive execution plans, pluggable databases, improvements on partitioning, identity columns, etc.

IOUG: Do you have any advice for novices in this industry?

Atil: The IT industry is constantly evolving and changing, but we have significant opportunities to adapt to this evolution. It's easy to reach valuable information through the Internet, and the Oracle community likes to share its knowledge. Of course, these can be also handicaps because they create endless competion. If you don't want to adopt new technologies, you stay out of the competition.

So my only advice for novices is that they should spend time on improving themselves. They will see that their personal return of investment is higher than anything else. You may ask how they can do it. They should join Oracle events, follow the publications, read blogs and write blogs (you learn while you share), and, lastly (but maybe the most important one) they should become an active member of user groups.

IOUG: Now that you're an Oracle ACE, what's next?

Atil: One of my goals is to become an Oracle ACE Director.

IOUG: Do you have any advice for IOUG members for their own careers?

Atil: Well, I think I already answered this question. IOUG members are one step ahead because they are already a part of one of the biggest Oracle user group!

IOUG: Is there anything else you'd like to add?

Atil: I would like to thank IOUG and the *SELECT Journal* editors for including my Oracle ACE profile in this issue.

Gokhan Atil Presents...

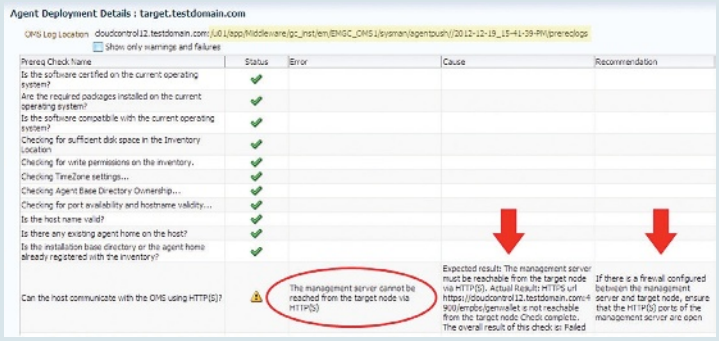
How to Troubleshoot Deploying Oracle Enterprise Manager Cloud Control 12c Management Agents

With every new version, Oracle Enterprise Manager provides more user-friendly experiences to the administrators. It is not just about having a better user interface; the new Enterprise Manager has simplified procedures for frequently used features such as deploying management agents. Even so, you may still get unexpected errors while deploying agents; and I'll try to explain some best practices to prevent errors and how to troubleshoot them.

There are several methods to deploy agents:

- PUSH Method: This is the recommended and easiest way to perform mass agent deployment. You can deploy agents using the "add host" wizard on the Enterprise Management web console. SSH should be running on both the remote host and OMS server if you want to use this method (you need to install cygwin to Windows servers). If the agent install user does not have SUDO privilege, you need to run root.sh script as root after you deploy the agent.
- Using AgentPull script (PULL Method): You can download the AgentPull script from OMS, create a response file and then use it to download and install the agent using AgentPull script.
- Using agentDeploy Script: You can create an agent image using emcli (Enterprise Management Command Line Interface) and then copy this image to remote host, create a response file and run the agentDeploy script to install the agent. This method is usually called as "silent" method, although RPM and PULL methods are also silent methods.
- Using RPM File: It's very similar to the agentDeploy method, but in this case you use EMCLI to create the agent image as RPM package. You copy it to the remote host and use rpm to install the agent.

All of these methods provide some level of prerequisite checks but, as I see, the best prerequisite checks are done by the "add host" wizard. It also provides "useful recommendations" to fix the problem.



In my experience, if you satisfy prerequisites, you have a high chance to deploy the agent successfully, so it's the key point of agent deployment.

If any errors occur while deploying the agent, you get an explaining error message in the progress page. You may search it on My Oracle Support and, in most cases, this would be enough to solve the problem, but if you need further inspection, you may read the log files.

If you use the PUSH Method ("add host"), all the agent deployment logs are available at:

$OMS_HOME/sysman/agentpush/*time_stamp*/*

On the PUSH method, you can also check the logs on remote host while deploying agents. A temporary folder under the AGENT_HOME is created. Log and errors are also stored there, but this folder will be deleted after the deployment is done:

$AGENT_HOME/*TMP_*time_stamp*/*.log

Other deployment methods than the PUSH method, log files will be located in where the agent image files are extracted.

Now, let's take a look at some common problems:

Connection problem between OMS and remote host

You should check if the host names of OMS and remote hosts can be resolved via DNS (or /etc/hosts file) by pinging OMS from remote host (and vice versa). Then you can check if you can access to OMS port. "telnet hostname port" would be useful for it.

You can learn the ports used by Enterprise Manage by running "emctl status OMS -details" on the OMS server.

Missing permissions on files or folders

Be sure that you gave permission to the agent install user, and fix the permissions by chmod or the ownership by chown.

Missing permissions on the Oracle inventory

Be sure that you have read, written and executed permissions on oraInventory on the remote host and that the Oracle Inventory (oraInventory) is not in a shared location.

Wrong agent registration password

If you do not remember the agent registration password you set while installing Enterprise Manager, you can create additional registration password through Cloud Control console.

Missing packages

You can install missing packages with using YUM or RPM tools. As you know, Oracle provides public yum server to help users to install required packages: **http://public-yum.oracle.com/**.

Oracle Enterprise Manager Cloud Control is a well-documented product. For prerequisites, agent deployment methods or any other information about Enterprise Manager Cloud Control, do not forget to check **http://www.oracle.com/pls/em121/homepage**.

There is also a very useful MoS document about troubleshooting management agent installations: ID 1396675.1.