Sharon combines her technology background with her Regis degree to help empower her own community. How will you be influential?

The School of Computer & Information Sciences in Regis University’s College for Professional Studies offers 4 undergraduate and 6 graduate degrees that immerse students in the latest technologies and software. Students benefit from our state-of-the-art learning environments, experienced faculty and strong industry relationships.

Busy adults choose Regis University for the academic rigor and our College for Professional Studies for the flexibility.

> Regionally accredited and nationally ranked
> Accelerated 5- and 8-week classes
> Online and campus-based learning
> Multiple start dates throughout the year

Volunteer for Training Days 2013!
Room Ambassadors Receive Full Registration*
Registration Ambassadors Receive 1/2 Registration Fee

For questions about Training Days, contact trainingdaysdir@rmougu.org.

February 12 & 13, 2013, with University Days on February 11
Information & Updates Available On http://www.rmougu.org

*Excludes University Days
Contact NewsletterDir@rmougu.org to Volunteer
On the Cover:
Mike Landrum’s photo of the Flat Irons after last February’s snow storm. Mike has been in the database business since 1985 and has been working exclusively in Oracle as a PL/SQL developer since 1996. He is currently enjoying the roll of data Tsunami at I-Behavior. You may see more of his photos at www.hikinmike.us/2012.
There are so many wonderful things about living in Colorado: hiking, biking, camping, fishing, skiing, snowboarding, snowshoeing, high technology, continuing professional education. All of these things are world-class things to do here.

It’s kind of funny, but the RMOUG “Training Days” conference started off 20 years ago as a one-day affair at the US West Training Center (as it was called then) in Lakewood. It was a big hit, and grew rapidly. Many of you reading this were there; I wasn’t. I missed the first RMOUG Training Day, but I presented at the second one. And I’ve only missed one since.

Then, around 1996, the conference outgrew one day and expanded into a two-day format, moving to a larger venue -- the Holiday Inn Conference Center at I-70 and Chambers. Within another two years, the “Training Days” conference had grown large enough to require the largest venue in the state -- the Colorado Convention Center.

The future was so bright through the end of the millenium, as attendance at the conference exceeded 1,200 people in 2000. The following year, RMOUG began pushing for “2001 in 2001”, but that was when the dot-com boom burst and the dot-bomb slide began, so while we never made it to 2000+ attendees, “Training Days” has continued to attract between 750–900 attendees steadily, an awesome turnout for this beautiful little old cowtown nestled between the prairie and the mountains.

All through this period, registration for the two-day conference stayed around $250, topping out at a little over $300 for late registration for non-members. Slowly, more and more speakers from around the country and around the world began augmenting the top-notch speakers, home-grown right here in Colorado.

In 2007, the Colorado Convention Center, gearing up for the expansion resulting in the doubling of its size, changed the game when it suddenly decided to charge RMOUG for the set-up day previous to the two-day conference, amending the contract. As a result, room expenses increased by 50%, blowing the budget. That year, RMOUG decided to increase the conference by a half-day by introducing the University sessions, four-hour afternoon seminars and workshops for a nominal fee of $112.50. Just like that, we now had a 2.5 day conference, and it was bigger, better, and more exciting than ever.

Oracle TechNet (OTN) joined the party last year, during TD2012, adding a new free Oracle-sponsored program called “OTN Developer Day” alongside the University sessions, comprised of a series of half-day seminars and workshops geared to Java, Forms, ADF, APEX, and PL/SQL developers. Many of the new attendees at the OTN Developer Day learned for the first time of the Training Days conference, and joined that too. And so the Training Days conference grew yet again, and became even more vibrant and varied.

This year, the conference is once again two-and-a-half days, packed with educational value, starting with a wide variety of half-day seminars on Monday, February 11, and continuing the main two-day Training Days 2013 conference on Tuesday and Wednesday, February 12-13 with 156 hour-long technical sessions.

Yes, that’s one hundred and fifty-six hour-long sessions, covering thirteen separate rooms over twelve time-slots over two days. The registration fees have increased a bit, starting at $275 for early registration for RMOUG members and topping out at a maximum of $495 for late registration for non-members.

Included in those 156 seminars will be presentations on Oracle database, application development, database administration, data warehousing and business intelligence, E-Business Suites, PeopleSoft, and Hyperion. Additionally, there will be over a dozen presentations focusing on MySQL, and this year, for the first time, a half-dozen presentations on SQL Server for the Oracle professional by some of the top SQL Server experts in the nation. As the breadth and scope of our profession expands, so too does RMOUG.

The real eye-openers are the six 4-hour University sessions on Monday, still at a low rate of $112.50, running concurrently with the Oracle-sponsored free OTN Developer Day, but wait! There’s more: Maria Colgan, senior director and lead developer of the Oracle cost-based optimizer, is providing one of the free Oracle-sponsored 4-hour seminars within OTN Developer Day. Additionally, Cary Millsap of Method-R is presenting his advanced top-notch 4-hour training seminar, “Mastering Oracle Trace Data”, for a fee of $700 on Monday as well.

All told, Colorado is going to be the epicenter of the Oracle world during the week beginning Monday, February 11. The very best and brightest in the world are coming to town, joining the best and the brightest who are already here. RMOUG “Training Days” is consistently ranked as one of the top six Oracle conferences in the world, alongside Open World, Collaborate, UKOUG, Kaleidoscope, and Hotsos Symposium. All of these conferences are the top in our field, and RMOUG Training Days is the most accessible and least costly of all.

Please join us at RMOUG Training Days 2013, from Monday through Wednesday, February 11-13, at the Colorado Convention Center. Join the experts at OTN, the RMOUG University sessions, Cary Millsap from Method-R, and more than 80 top minds from around the world. Bring your colleagues, bring your co-workers, and experience first-hand the very tip-top best in professional continuing education anywhere in the world.

Please check out the Training Days menu at www.rmoug.org/training, see the amazing agenda in store, and register now. Also, follow RMOUG on Facebook, LinkedIn, and Twitter, using the icons on the www.rmoug.org website.
Stan Yellott Scholarship Fund

Eligibility Requirements

- Registered or the intent to register at an accredited post secondary educational institution in the United States
- Minimum GPA 2.5 from a current official transcript
- Currently enrolled in a computer science/information technology class and/or intends to enroll in a computer science the following term

Award notification will be given within 45 days following application deadlines. Upon acceptance of the scholarship, additional information may be required.

For Details, Visit the RMOUG Website
www.rmoug.org

RMOUG Scholarship Mission

To provide educational opportunities to members of the organization about the information technology industry in general, and in particular the technology of Oracle Corporation to include databases, storage, networking and application development, specifically the products and services of the Oracle Corporation.

To collect, post and distribute information about Oracle technologies and other related technologies to members.

To provide members with the ability to help their peers maximize their knowledge and skills working with products in information technology and in particular Oracle products.

To provide a consolidated channel of communication, conveying needs, concerns, and suggestions, for members of the organization, to Oracle Corporation and other vendor corporations involved with Oracle related technology.

To encourage members to present their information technology experiences using Oracle and other products and services.

To provide a consolidated channel of communication between members of the RMOUG and other communities in related information technology industries.

To promote educational opportunities for students of information technology through directed funding and services for educational purposes.

RMOUG is committed to supporting others in the pursuit of technical knowledge.

The Scholarship Fund started in 2001 to encourage future IT professional in their efforts to broaden their knowledge. In 2007, RMOUG voted to rename the scholarship fund to honor the memory of Stan Yellott. Stan was a long time member of RMOUG where he supported the user community by serving on the RMOUG board. Stan focused on expanding Oracle educational opportunities. Stan’s vision was to include high school and college students as the next generation of IT professionals.

Become A Member For These Outstanding Benefits

- Quarterly Education Workshops
- Special Annual Training Days Rates
- Database Labs
- List Server Subscriptions
- SQL>Update Magazine

www.rmoug.org/member.htm
Creating Oracle ADF 11g Bounded Task Flow Train

by Julie Johnson

Overview

In this tutorial, we’re going to create a simple process train that allows the user to create a new employee, stepping through several pages through the process. Since an Employee record contains many fields, it is more user-friendly to break up data entry into several steps. The first page is for collecting the contact information, the second page is for salary data, the third is for the remaining information, and the last page will be the review page where the user can decide whether to save or cancel.

Below are some screenshots of what the end product will look like.

We have a page that displays the employee records.

When the user clicks on the New Employee button, the process train is launched, displaying the first page in the train.

Clicking on the Salary button or Next button navigates to the Salary page.

The following page is a read-only form that displays all of the data the user entered. Clicking on the Save Changes buttons commits, whereas clicking on Cancel rolls back any changes.

After saving changes, the user now sees the new record in the Employee table.
The Model Layer

Since this application is based on the HR schema's EMPLOYEES table, we need to create the necessary EmployeesVO view object to expose as a business service. We are using an updatable EMPLOYEES view object, which means you will have a corresponding EmployeeEO (Entity Object).

We will also have read-only view objects for the JOBS, EMPLOYEES, and DEPARTMENTS tables (JobsROVO, EmployeesROVO, and DepartmentsROVO, respectively to support the LOVs (Lists Of Values) for the DepartmentId, JobId, and ManagerId attributes.

Since we want to have a primary key automatically generated for the new employee, we set the EmployeeEO's EmployeeId attribute data type to Integer and set a default value to the following expression:

\[
\text{(new oracle.jbo.server.SequenceImpl("EMPLOYEES_SEQ",adf.object.getDBTransaction()).getSequenceNumber())}
\]

We can default the EmployeeEO's hiredate to the current date by setting it to the following expression:

\[
\text{adf.currentDate}
\]

Creating the Bounded Task Flow

We now need to create the bounded task flow for creating a new employee.

1. Create a new bounded task flow (New -> JSF/Facelets -> ADF Task Flow). Give it a name such as “new-emp-taskflow.xml”. Deselect the "Create with Page Fragments" check box if you want the pages to be stand-alone pages (Alternatively you could choose to use page fragments, then include them as page regions). Be sure to check the “Create Train” box.

2. With the new task flow visible in the main window, drag onto it the EmployeeVO's CreateInsert operation, as well as four view activities, naming them ‘contact’, ‘salary’, ‘otherInfo’, and ‘review’. Also drag two task flow return activities, naming them ‘commit’ and ‘rollback’. Note that the CreateInsert activity is highlighted in green. This means it is the default activity, which means it is the main entry point into the task flow. This activity is responsible for creating a new record and placing it in the iterable record set.

3. Since we specified this task flow to be a train, the views are automatically connected with dotted lines. Create control flow cases from the CreateInsert to the contact activity (call it ‘new-Emp’). Click on the CreateInsert activity. In the property Inspector set the Fixed Outcome property to ‘newEmp’. It should be the only option in the drop-down menu. This simply means that when the CreateInsert activity is complete, the next step is to automatically navigate to the ‘contact’ view activity.

Testing in the Business Components Browser

You can test the functionality of the EmployeesVO instance in the Business Component Browser before using it in a page – simply right-click on the Application Module from the Application Navigator, then hit the green arrow to run. Double-click on the EmployeesVO1 instance, then click on the green plus icon to add a new record. After entering the data, click on the Commit button to save your changes.
4. Create two more control flow rules, each originating from the 'review' view activity and pointing to task flow return activities ('commit' and 'rollback'). Don't worry about the red X's on the view activities. This just means we haven't created the pages yet!

5. Since we want a new transaction to begin when we enter this task flow, click in the background of the task flow, then look at the Property Inspector for the task flow definition. Set the Transaction property to 'Always begin new transaction'.

6. We want our train component to display text for each of the train stops – 'contact', 'salary', 'otherInfo', and 'review'. To accomplish this, select the 'contact' view activity. In the Structure window, you should see it highlighted. Expand it until you see the train stop. Insert inside the train stop a display-name, setting it to 'Contact' in the Property Inspector. Perform the same steps for the other view activities until it looks like this in the Structure Window:

---

**Creating a Page Template**

Each of the pages will have a similar look and feel. They will all contain a Train and Train Button Bar component. Creating a page template for your pages is faster and more scalable. To create the template:

1. Go to New -> ADF Page Template. Give it a name, such as 'trainTemplate1.jsf'. Select 'facelets' as the document type, and don't use any quick start layouts.

2. Add a facet definition, calling it 'body'. This will be used in the template as a placeholder for editable content for the pages that use the template.

3. Click on the Attributes tab. Add an attribute called 'customTitle', giving it a default value of 'Custom Title'. We can use this attribute wherever text can be used, such as a panel header title.

Let's now lay out the components on the page. Place a Panel Stretch Layout on the page. Place a Train component in the Top facet and a Decorative Box in the Center facet. In the Decorative Box's top facet, place a Train Button Bar component. Set the Train and TrainButtonBar's value to the train model: #{controllerContext.currentViewPort.taskFlowContext.trainModel}

In the DecorativeBox's Center facet, place another Decorative Box. In the second Decorative Box's Top facet, place a Panel Header component. Set the text to #{attrs.customTitle}. You can find this under Scoped Variables using the Expression Editor.

4. In the DecorativeBox's Center facet, place a FacetDefinition component. Select the Body facet that you defined earlier from the drop-down menu.

The template’s Structure Window should now look like this:
Create the Pages from the Template

Let’s create our pages now. From the task flow diagram, double-click on the ‘contact’ view activity. Create a page from the template:

The part of the page that is grayed out is the template, whereas the white area is the editable area as defined by the body facet you created. You can now override the Panel Header’s text by clicking on the Panel Header, then setting the text to ‘Contact’ in the Property Inspector.

Repeat those steps for the remaining three view activities, setting the headings to ‘Salary’, ‘Additional Information’, and ‘Review’.

Add Form Components to the Pages

1. To add contact form components to the Contact page, double-click on the page, then drag over the EmployeesVO instance, selecting ‘ADF Form’. Only select the EmployeeId, LastName, FirstName, Email, and PhoneNumber. Click on the Binding tab, then select the contactPageDef1 from the Structure Window. In the Property Inspector, set SkipValidation to ‘true’ – we don’t want validation kicking in yet because the next few pages contain required information!

2. For the Salary page, follow the same steps as for the Contact page. This time, only select the Salary and CommissionPct attributes.

3. For the Additional Info page, follow the same steps as before, only including the attributes that haven’t been used yet (HireDate, JobId, ManagerId, DepartmentId). Do not skip validation on this page, as we are on the last page that collects data.

4. For the Review page, drag over the EmployeesVO instance to create an ADF Read-only Form. Add a Panel Group (horizontal layout) before the form. Add two Buttons – ‘Save Changes’ and ‘Cancel’ with Action properties of ‘commit’ and ‘rollback’, respectively. After either one of these buttons is pressed, navigation returns to the caller of this task flow.

Create a Page that Calls the Task Flow

Let’s create a page that displays the existing employees. We’ll include a button on the page that navigates to the bounded task flow. To do this:

1. Go to the adfc-config task flow. This is the default, unbounded ‘bootstrap’ task flow that comes with every ADF Fusion ViewController project.

2. Drag onto the diagram a view activity, calling it ‘empTable’. From the Application Navigator drag the new-emp-taskflow onto the diagram. This creates a task flow call activity that references our bounded task flow.

3. Create a control flow case from the view activity to the task flow call. Set the outcome text to ‘newEmp’. Don’t worry about the warning on the empTable view activity – this will go away once we have created the page.
4. Double-click on the empTable view activity to create the page. For now, let’s just create a page not based on a template.

Place a Panel Stretch Layout onto the page, then add a Panel Header to the Top facet. Place a Decorative Box on the Center facet, adding an ADF Read-only table to the Box’s Center facet. Place a Panel Group Layout (horizontal layout) and Command Button to the Decorative Box’s Top facet. Set the button’s text to ‘New Employee’ and set the Action property to ‘newEmp’ (it should appear in the drop-down menu).

Test Your Application

You can now test your application by running the empTable.jsf page!

Julie Johnson is the founder and CEO of Firebox Training, based in Golden, Colorado. She has taught and consulted in the IT industry internationally for over ten years with a focus on Oracle ADF, Oracle Apex, Java EE, and OBIEE. You can find more IT tutorials on the company website at http://www.fireboxtraining.com/blog.

<table>
<thead>
<tr>
<th>Breakfast</th>
<th>Discounted Ad Rate</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>$350.00</td>
<td>$175.00</td>
<td>$525.00</td>
</tr>
<tr>
<td>$350.00</td>
<td>$312.50</td>
<td>$662.50</td>
</tr>
<tr>
<td>$350.00</td>
<td>$500.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>$350.00</td>
<td>$625.00</td>
<td>$975.00</td>
</tr>
<tr>
<td>$350.00</td>
<td>$750.00</td>
<td>$1,100.00</td>
</tr>
</tbody>
</table>

Contact Carolyn Fryc - Programs Director - 720-221-4432 - cfryc@orsportal.com

Help RMOUG Members and Receive Recognition in An Upcoming Issue of SQL>Update
The era of DATA has arrived

Database Services Designed for Today’s Enterprise

ON-SITE & OFFSITE
Expertise when and where you need it

ON-SHORE & OFF-SHORE
The right resource at the right time

24X7 SUPPORT
We don’t sleep, so you can rest easy

ITIL & TIERED DELIVERY
Enterprise-grade service delivery projects

SERVICE MANAGEMENT
Bringing the right people, process and tools to your data

MONITORING & TOOLS
Database automation and management, built from the ground up

datAvail
www.datavail.com
Oracle Password Store
Never Have Passwords in Scripts Again

by Mike Messina
Senior Managing Consultant, Rolta Solution, TUSC Infrastructure Services

Summary
Starting with Oracle 10g, Oracle provides a secure way for processes to make connections to a database. The Oracle Password Store Wallet allows for database connections made via scripting processes such as backup jobs, maintenance jobs and system batch processes and will work with any database user account on any server or client machine. The database connection can be made without having to code passwords into scripts or environment variables that can compromise the security of the password and the password store capability does not require any additional license such as the Advanced Security Option.

Create TNS Location
It is recommended that the standard wallet location consist of a location identified by the TNS_ADMIN environment variable and the location should be located outside the Oracle binary home location (ORACLE_HOME). Since it is a location outside the ORACLE_HOME binaries, individual OS accounts can have different locations allowing many wallets for many users on the same system connecting to the same databases.

Unix
$ORACLE_BASE/network/admin

Windows
%ORACLE_BASE%\network\admin

Create Standard Wallet Locations
Within the TNS_ADMIN location wallets can exist in 2 forms and can not be co-located. Therefore an encryption and Authentication location should be created to separate the wallets. As a best practice create a location for the encryption wallet as well as the password store wallet.

Location for Encryption Wallet
Unix
$ORACLE_BASE/network/admin/encrypt

Windows
%ORACLE_BASE%\network\admin\encrypt

Location for Password Store Wallet
Unix
$ORACLE_BASE/network/admin/authent

Windows
%ORACLE_BASE%\network\admin\authent

Create Wallet For Password Store
a. cd $ORACLE_BASE/network/admin/authent
b. mkstore –wrl $ORACLE_BASE/network/admin/authent –create
c. Enter password for wallet (will have to be entered 2 times for confirmation)*

* Creates 2 files ewallet.p12 and cwallet.sso

Create Credential(s) For Password Store Wallet
For Example RMAN Catalog
For connections to an RMAN catalog when executing RMAN backups that have an RMAN catalog, the connection to the RMAN catalog is usually a remote connection and requires a password in the script processes that make the connection. This is bad for security purposes; therefore, a password store wallet is a great solution.

Format
mkstore –wrl $ORACLE_BASE/network/admin/authent
–createCredential <db>_<username> <username> <user password>

Example
mkstore –wrl $ORACLE_BASE/network/admin/authent
–createCredential rman_rman rman <rman user password>

* should see something like
Create credential oracle.security.client.connect_string1

For A Target Database
Backup processes are one good use for wallets, but usually DBAs and Developers have many needs to run processes against an Oracle database requiring authentication. A password store wallet is a good way to keep passwords out of the batch process files.
Format
mkstore -wr1 $ORACLE_BASE/network/admin/authent -createCredential <db>_<username> <username> <user password>

Example
mkstore -wr1 $ORACLE_BASE/network/admin/authent -createCredential mydb_system system <system user password>

* should see something like

Create credential oracle.security.client.connect_string2

Set The Client To Be Able To Use The Password Store Wallet

For the Oracle Client to utilize the password store wallet that was created, we need to create a sqlnet.ora file in the location setup and assigned to TNS_ADMIN. The client will utilize this file, allowing the client to locate the password store wallet and check the wallet for the credentials based on the TNS connect string. Besides identifying the location of the wallet, the sqlnet.ora will ensure that the client reads the wallet when facilitating a connection request.

Unix server sqlnet.ora
ENCRYPTION_WALLET_LOCATION=
(SOURCE=
(METHOD=FILE)
(METHOD_DATA=
(DIRECTORY= /u01/app/oracle/network/admin/encrypt)
)
)
WALLET_LOCATION=
(SOURCE=
(METHOD=FILE)
(METHOD_DATA=
(DIRECTORY= /u01/app/oracle/network/admin/authent)
)
)
SQLNET.WALLET_OVERRIDE = TRUE
SSL_CLIENT_AUTHENTICATION = FALSE
SSL_VERSION = 0

Restart Or Reload The Database Listener

Due to the changes made to the sqlnet.ora, for the changes to take affect so the wallets can be utilized, it is recommended to bounce or reload the listener.

lsnrctl reload
or
lsnrctl stop <listener name>
lsnrctl start <listener name>

Test The Wallet Credentials To Login

Now that the wallet password store has been created, the sqlnet.ora and tnsnames.ora is setup to utilize the wallet. We need to test and make sure the wallet password store is functioning as expected.

sqlplus /@RMAN_RMAN
sqlplus /@MYDB_SYSTEM
sqlplus /@MYDB_SYS
rman target=/@MYDB_SYS catalog=/@rman_rman

Michael Messina is a Senior Managing Consultant with Rolta Solutions, TUSC Infrastructure Services. An Oracle ACE, Oracle Certified Professional, Oracle RAC Administrator Certified Expert and Exadata Implementation Specialist with nearly 20 years experience with the Oracle Databases. He has contributed to several chapters in the newest Oracle Database 11g Release 2 Performance and Tuning Tips & Techniques from Rich Niemiec. Michael has presented at Open World, RMOUG Training Days, IOUG Collaborate as well as many local and regional users groups throughout the U.S. Michael’s experience includes Oracle and MySQL database administration and implementation, system and infrastructure implementation. Michael has led several performance improvement, maintenance, and implementation projects for large highly available systems.

Michael Messina is a Senior Managing Consultant with Rolta Solutions, TUSC Infrastructure Services. An Oracle ACE, Oracle Certified Professional, Oracle RAC Administrator Certified Expert and Exadata Implementation Specialist with nearly 20 years experience with the Oracle Databases. He has contributed to several chapters in the newest Oracle Database 11g Release 2 Performance and Tuning Tips & Techniques from Rich Niemiec. Michael has presented at Open World, RMOUG Training Days, IOUG Collaborate as well as many local and regional users groups throughout the U.S. Michael’s experience includes Oracle and MySQL database administration and implementation, system and infrastructure implementation. Michael has led several performance improvement, maintenance, and implementation projects for large highly available systems.
An Introduction to Oracle Heterogeneous Services

by Steven Wales, Addons, Inc.

Introduction

Oracle Heterogeneous Services allows an Oracle database system to connect seamlessly to a non-Oracle system. Oracle Transaction Gateways specific to a targeted third party database system are available under separate licenses from Oracle. However, starting in 8.1.6, Oracle introduced the ability to connect to non-Oracle (Heterogeneous) systems through ODBC and OLE DB standards.

Up until Oracle 10g there were published Transparent Gateways for targeted non-Oracle databases and then a Generic Connectivity Agent for ODBC and OLE DB.

In Oracle 11g, the Generic Connectivity Agent became the Gateway Agent for ODBC.

Using the Generic Connectivity Agent / ODBC Gateway as opposed to one of the product specific Transparent Gateway Agents also limits functionality in some aspects. When deciding on what type of connection to make, the full impact of the requirement should be examined against the feature sets offered by each agent when deciding how to connect.

This is something that should be considered when deciding which connectivity method to use in your environment.

This article will examine the ODBC Gateway Agent and how to make an Oracle database communicate with Microsoft SQL Server via the ODBC Gateway Agent (and by extension, any ODBC compliant database system).

Definitions

There are some key definitions that need to be understood when dealing with Oracle Heterogeneous Services. These definitions are as defined by Oracle in “Oracle 11g Database Administrator’s Guide” (Oracle 11g Release 2 Publication E25494-02, December 2011).

Distributed Database System – Allows applications to access data from local and remote databases.

Homogenous Distributed Database System – A network of two or more Oracle databases that reside on one or more machines. This configuration can be made of databases using many different versions of the Oracle database software, however, each database must understand all extensions used in applications. Using an Oracle 11g specific function on an Oracle 9i database will not work.

Heterogeneous Distributed Database System – A network of two or more databases that reside on one or more machines where at least one of the database systems is a non-Oracle system. To the application, the heterogeneous distributed database system appears as a single, local, Oracle database; the local Oracle database server hides the distribution and heterogeneity of the data.

Distributed Databases – A set of databases in a distributed system that can appear to applications as a single data source.

Distributed Processing – The operations that occur when an application distributes its tasks among different computers in a network. For example an application typically distributes front end presentation tasks to the client computers and allows a back end database server to manage shared database access.

Heterogeneous Services – Heterogeneous Services is an integrated part of the Oracle Server and also the enabling technology behind Oracle Transparent Gateway products.

Transparent Gateway Agents – For each non-Oracle system you use, Heterogeneous Services can use a transparent gateway agent to interface with the specific non-Oracle system. Each agent is specific to the non-Oracle system. Transparent gateway agents are separately licensed.

Generic Connectivity – As an alternative to using a transparent gateway agent, it is possible to connect to a non-Oracle system using either a Heterogeneous Services ODBC Agent or a Heterogeneous Services OLE DB Agent, both of which are included as a standard feature in Oracle. Generic Connectivity offers a much reduced functionality set than what is offered with a Transparent Gateway Agent. In Oracle 11g, the Generic Connectivity Agent is also treated as its Gateway Agent.

Features of Heterogeneous Services

Some of the key features of Heterogeneous Services are included in the table on the following page:

There are, of course, other features in the Heterogeneous Services feature set, but these will not be covered in this article. Refer to Oracle system documentation for specific information about all product features.

Connecting To a Database

There are two different ways to make a database connection.

A Direct Connection is made when a client connects to a server and accesses information from a database on that server.

An Indirect Connection is made when a client connects to a server and accesses information from a database on a different server.

For example, consider the following two queries:

```sql
select * from dept;
select * from emp@sales;
```
Distributed Database System | Allows applications to access data from local and remote databases.
---|---
Homogeneous Distributed Database System | A network of two or more Oracle databases that reside on one or more machines. This configuration can be made of databases using many different versions of the Oracle database software, however, each database must understand all extensions used in applications. Using an Oracle 11g specific function on an Oracle 9i database will not work.
Heterogeneous Distributed Database System | A network of two or more databases that reside on one or more machines where at least one of the database systems is a non-Oracle system. To the application, the heterogeneous distributed database system appears as a single, local, Oracle database; the local Oracle database server hides the distribution and heterogeneity of the data.
Distributed Databases | A set of databases in a distributed system that can appear to applications as a single data source.
Distributed Processing | The operations that occur when an application distributes its tasks among different computers in a network. For example an application typically distributes front end presentation tasks to the client computers and allows a back end database server to manage shared database access.
Heterogeneous Services | Heterogeneous Services is an integrated part of the Oracle Server and also the enabling technology behind Oracle Transparent Gateway products.
Transparent Gateway Agents | For each non-Oracle system you use, Heterogeneous Services can use a transparent gateway agent to interface with the specific non-Oracle system. Each agent is specific to the non-Oracle system. Transparent gateway agents are separately licensed.
Generic Connectivity | As an alternative to using a transparent gateway agent, it is possible to connect to a non-Oracle system using either a Heterogeneous Services ODBC Agent or a Heterogeneous Services OLE DB Agent, both of which are included as a standard feature in Oracle. Generic Connectivity offers a much reduced functionality set than what is offered with a Transparent Gateway Agent. In Oracle 11g, the Generic Connectivity Agent is also treated as its Gateway Agent.

If we assume that we connect to the HQ database and query the DEPT table in that database on the local machine, that is a direct connection. Then, while connected to the HQ database on the local machine, you also want to query the EMP table in the remote SALES database, then that is an indirect connection.

**Database Links**

A database link is a central concept for accessing a remote database. It allows a connection to be made between an Oracle database and another remote database (both Oracle and non-Oracle).

A database link is a one way tunnel between two databases. If database A creates a database link to database B, then database A can query database B, but database B cannot use the same link to query database A. Database B would need to create its own database link pointing to database A in order to be able to query from it.

It is possible to create a synonym to make the query from the remote database appear to be a local table. The synonym provides a layer to hide the appearance of accessing a remote database. When dealing with synonyms in this manner be sure that local objects are not affected by the use of the synonym, such as having a local EMP object and a synonym to a remote EMP object.

There are many security options for setting up database links from private and public links to the manner in which the database links use authentication on the remote systems, but this is outside of the scope of this paper. Database links are well documented in the product documentation and the reader should review there for more information on this particular topic.

A database link enables standard insert, update, delete and select statements to be executed on the remote database system. However, there are many things that cannot be done across the link, including but not limited to:

- Grant privileges
- Execute DESCRIBE on some remote objects (tables, views, procedures and functions are allowed)
- Analyze remote objects
- Grant roles to users
- Define or enforce referential integrity

**Transactional Consistency**

It is just as important in a distributed environment as it is in a local environment to maintain transactional consistency in your databases.

A transaction begins with the first executable statement and ends when it is committed or rolled back.

A remote transaction involves statements executed in a local database that updates only tables in a remote database.

A distributed transaction is a transaction that updates data in two or more distinct nodes of a distributed database system as a part of a single transaction.

In order to control the consistency of these updates in a non-local transaction, Oracle uses a Two Phase Commit mechanism to guarantee the transaction.

When performing a distributed transaction, it is important that all nodes in the transaction perform the same function – every node commits or every node rolls back the transaction.

At a high level, the global transaction coordinator will ask every node in the transaction to prepare to commit – they make a promise to either commit or rollback the transaction. One the coordinator has an acknowledgement from all transaction nodes, it then
asks them to commit. If it is not possible for all nodes to commit, then all nodes will rollback.

Distributed transactions are not supported under the Generic Connectivity Agent. Transactions need to be for one node only when using the Generic Agent.

The Two Phase Commit mechanism is fully documented in the Administrator’s Guide and further reading can be had in that book.

Putting It All Together

With the underlying concepts explained, how do we make an Oracle database talk to a non-Oracle database?

First, let’s lay out some ground rules and expectations based upon the specific environment I was working with the first time I set this up in my production environment:

1. Oracle system is Oracle 10g (10.2.0.4), on Windows XP (Server A)
2. Non-Oracle system is Microsoft SQL Server 2005 Service Pack 3 on Windows Server 2003 (Server B)
3. Oracle system contains a time keeping application that needs to validate work orders, project codes and account codes in real time against the primary financial system. This system resides on Server A.
4. Financial system resides on SQL Server on Server B.

It should be noted that these instructions were generated around an Oracle 10g database. Some changes were made in Oracle 11g and these will be noted as necessary. In 11g, the “Generic Connectivity Agent” nomenclature was changed to “Gateway Agent for ODBC” and has its own documentation.

Step 1

The first step, for completeness, is to create the SQL Server database to work in. Execute the following on Server B:

```
CREATE DATABASE [FinanceDemo] ON PRIMARY
( NAME = N 'FinanceDemo', FILENAME = N'E:\MSSQL\DATA\FinanceDemo.mdf' ,
  SIZE = 5120KB , FILEGROWTH = 1024KB )
LOG ON
( NAME = N 'FinanceDemo_log', FILENAME = N'E:\MSSQL\DATA\FinanceDemo_log.ldf' ,
  SIZE = 1024KB , FILEGROWTH = 10%)
```

Step 2

On Server B, create objects in the database to query against. use FinanceDemo

```
GO
CREATE TABLE WORK_ORDER (WO_NO CHAR(8 ), WO_DESC CHAR (40));
GO
CREATE TABLE PROJECT (PROJ_NO CHAR(8 ), PROJ_DESC CHAR (40));
GO
CREATE TABLE ACCOUNT_CODE (ACCT_NO CHAR(8 ), ACCT_DESC CHAR (40));
GO
```

Step 3

Create credentials on the non-Oracle database, and grant appropriate permissions. For now, we’ll just grant SELECT on the three objects.

```
CREATE LOGIN [findemo] WITH PASSWORD =N 'findemo' ,
DEFAULT_DATABASE = [FInDemo] , DEFAULT_ LANGUAGE = [us_english],
CHECK_EXPIRATION = OFF , CHECK_POLICY = OFF; 
GO
CREATE USER [findemo] FOR LOGIN [findemo] 
WITHDEFAULT_SCHEMA= [dbo]; 
GO
GRANT SELECT ON WORK_ORDER TO [findemo]
GO
GRANT SELECT ON PROJECT TO [findemo]
GO
GRANT SELECT ON ACCOUNT_CODE TO [findemo]
GO
```

Step 4

Ensure that appropriate ODBC drivers exist on Server B. For some databases, such as MySQL, the ODBC drivers must be installed separately to the database software. Create an ODBC connection on Server B connecting to the non-Oracle database using the credentials defined. The ODBC Administrator is accessed from Start / Control Panel / Administrative Tools / Data Sources (ODBC). Select the System DSN tab and click “Add” and follow the prompts, being sure to select the appropriate ODBC driver. Test the connectivity to the target database when done to ensure it works.

Step 5

For this example, for simplicity’s sake, we will be ensuring that our Oracle database has global_names set to false. Oracle recommends setting global_names to true since many advanced features require it. Also, when global_names is true the name of the database link that connects to a remote database must also be the same as the global name of the database. If global_names is false, then a database called sales.acme.com could have a database link named “MY_LINK”. For our example, we’ll be ensuring that global_names is false on the Oracle Database on server A.

Check the parameter setting by querying:

```
select * from v$parameter where name = 'global_names';
```

If the parameter is not false, then set it:

```
alter system set global_names=false scope=both;
```

Step 6

On Server B, the Oracle Database software needs to be installed with enough options to cover ODBC and to be able to configure and start a Listener. For Oracle 10g and earlier, this usually means installing the database engine. For Oracle 11g, there is an option in the Oracle Universal Installer as a part of the gateways install to install the Gateway Agent for ODBC.

Step 7

On Server B, the Heterogeneous Services Agent needs to be configured. Navigate to ORACLE_HOME\hs\admin and you will see a file called inithsodbc.ora. Copy this file to initDSNNAME.ora (in our example here, we create a file called initfindemo.ora). Find the line HS_FDS_TRACE_LEVEL and set the parameter
to OFF.

Find the line HS_FDS_CONNECT_INFO and set the parameter to the name of the ODBC DSN (findemo here).

It is important to note that the Generic Connectivity Agent was changed in Oracle 11g and the template file for that release is called initdg4odbc.ora. However the syntax in the file is unchanged.

The contents of the initfindemo.ora file will look like this:

```sql
# This is a sample agent init file that contains
# the HS parameters that are
# needed for an ODBC Agent.
#
# HS init parameters
# HS_FDS_CONNECT_INFO = findemo
HS_FDS_TRACE_LEVEL = off
#
# Environment variables required for the non-Oracle system
#
#set <envvar>=<value>
```

The Trace Level option turns off error tracing. In the event that error tracing is needed to debug connectivity issues, the log is stored in ORACLE_HOME/log.

**Step 8**
The Oracle Listener must be configured on Server B to point to the entry we just created in the previous step. Create an entry in listener.ora as follows:

```sql
SID_LIST_LISTENER=
 (SID_LIST=
  (SID_DESC=
   (SID_NAME=findemo)
   (ORACLE_HOME=c:\oracle\product\10.2.0\db_1)
   (PROGRAM=hsodbc)
  )
)
```

The SID_NAME tells the listener to look at the init file created previously. The Program parameter tells Oracle which driver to use, and it should exist in ORACLE_HOME/bin. The Oracle listener should be restarted on this server after this change is saved.

As noted in Step 7, with the changes made in 11g, the PROGRAM_NAME parameter would be dg4odbc, not hsodbc as shown.

**Step 9**
The tnsnames.ora file, on each Oracle system connecting to the non-Oracle system needs to be modified to tell Oracle where to look for the non-Oracle system.

In our example, this means that tnsnames.ora needs to be modified on Server A as follows:

```sql
FINDEMO =
 (DESCRIPTION =
 (ADDRESS_LIST =
  (ADDRESS = (PROTOCOL = TCP)(HOST = myserver.mycompany.com)(PORT = 1521)))
 (CONNECT_DATA = (SID = findemo))
(HS=OK)
)
```

This pretty much looks like any other tnsnames entry, except for the (HS=OK) parameter. This tells Oracle to use Heterogeneous Services when attempting to resolve this connection.

**Step 10**
Test your connection from the Oracle Server by executing “tnsping findemo”.

You would expect to see something like this:

```bash
C:\Documents and Settings\waless>tnsping findemo
TNS Ping Utility for 32-bit Windows: Version 10.2.0.5.0 - Production on 02-JAN-2012 13:51:50
Copyright (c) 1997, 2010, Oracle. All rights reserved.
Used parameter files:
C:\oracle\product\10.2.0\db_1\network\admin\sqlnet.ora
Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = myserver.mycompany.com)(PORT = 1521)) (CONNECT_DATA = (SID = findemo)) (HS=OK)))
OK (250 msec)
```

**Step 11**
The last step needed is to create a database link between the Oracle database and the non-Oracle database. It is important to note for SQL Server that the username and password should be in double quotes and the TNS alias should be in single quotes. Failure to set up the database link in this manner may generate an ORA-28500 error “connection from ORACLE to a non-Oracle system returned this message:”, followed by an error message from the gateway about invalid credentials.

Using the quotes will maintain the case of these key words when sent and not trigger failure in case sensitive remote systems.

```sql
create [public] database link findemolink
connect to “findemo” identified by “findemo” using ‘findemo’;
```

Obviously not everything has to be “findemo”. The username and password would be as you created and the database link name could be anything (since we set global_names to false). The “using” parameter needs to match the ODBC connection name defined on Server B.

**What Can We Do With It?**

Now that we have it all set up, what can we do with it? An initial test can be made to make sure we at least have a connection:

```sql
SQL> select * from project@findemolink;
```

---

**What Can We Do With It?**

Now that we have it all set up, what can we do with it? An initial test can be made to make sure we at least have a connection:

```sql
SQL> select * from project@findemolink;
```
We could go and insert a dummy project into the SQL Server database and then fetch it:

```
insert into project values ('PRJ0001', 'My test project');
```

And then query it from Oracle:

```
SQL> select * from project@findemolink;
PROJ_NO  PROJ_DESC
-------- ----------------------------------------
PRJ0001  My test project
```

If we try to update this project from Oracle, noting that we previously had granted only select access, we find that we don’t have permission:

```
SQL> update project@findemolink set proj_desc = 'Something different' where proj_no = 'PRJ0001';
update project@findemolink set proj_desc = 'Something different' where proj_no = 'PRJ0001'
* 
ERROR at line 1: 
ORA-28500: connection from ORACLE to a non-Oracle system returned this message: 
ORA-02063: preceding 2 lines from FINDEMOLINK
```

However, if we go and grant update on the project table on the SQL Server side and try again, we see it works perfectly:

```
SQL> update project@findemolink set proj_desc = 'Something different' where proj_no = 'PRJ0001';
1 row updated.

SQL> select * from project@findemolink;
PROJ_NO  PROJ_DESC
-------- ----------------------------------------
PRJ0001  Something different
```

One of the features of Heterogeneous Services was Data Dictionary translation. In SQL Server, there are no DBA_ tables. The data dictionary is represented in the dictionary via system views like sys.tables. Executing a query on the SQL Server side that tried to read DBA_TABLES would generate an error:

```
select * from sys.tables
Msg 208, Level 16, State 1, Line 1
Invalid object name ‘sys.tables’.
```

However, execute something similar via the Oracle side connection (column list shortened to fit the page) and you can see the tables that we created earlier:

```
SQL> select owner, table_name, num_rows
2  from dba_tables@findemolink;
```

<table>
<thead>
<tr>
<th>OWNER</th>
<th>TABLE_NAME</th>
<th>NUM_ROWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbo</td>
<td>ACCOUNT_CODE</td>
<td>0</td>
</tr>
<tr>
<td>dbo</td>
<td>PROJECT</td>
<td>1</td>
</tr>
<tr>
<td>dbo</td>
<td>WORK_ORDER</td>
<td>0</td>
</tr>
</tbody>
</table>

We also talked about data type transformation as a feature of Heterogeneous Services.

Let’s create a table in SQL Server, populate it with a date and then query it from Oracle.

On SQL Server, the setup looks like this:

```
create table datetest (date1 datetime);
grant select on datetest to findemo;
insert into datetest values (getdate());
```

```
SQL> select * from datetest@findemolink;
```

```
date1
-----------------------
2012-01-02 14:29:56.950
```

This value can be manipulated by altering the NLS_DATE_FORMAT parameter if we wish to see the date and time in a different way. This data is being seamlessly translated by the Heterogeneous Services Agent.

```
SQL> alter session set NLS_DATE_FORMAT='YYYY-MM-DD HH24:MI:SS';
Session altered.
SQL> select * from datetest@findemolink;
```

```
datel
-------------------
2012-01-02 14:29:56
```

Another of the features mentioned earlier is Pass-Through SQL. In order to achieve this the DBMS_HS_PASSTHROUGH package is used. This is not a physical package in the database but a conceptual package that can be invoked when using a Heterogeneous Services link.
The example below shows off several of the differences between a direct Oracle call and a pass-through call to the non-Oracle system.

1. In the native calls, note that the Oracle string concatenation format is used (||) but in the pass-through statements, the SQL Server string concatenation format is used (+).
2. When getting current date in Oracle, we need to SELECT from DUAL. In SQL Server, no FROM clause is needed.
3. The pass-through statements allow us to use native SQL Server functions, such as CONVERT (which is a much different function than Oracle's CONVERT and GETDATE()).
4. Note that when querying the “date1” column of table datetest, the column name needs to be double quoted. Column names are converted to uppercase in Oracle on table creation. This is not the case in SQL Server. Since the column name is lower case, we must tell Oracle to preserve the case and we do that by double quoting it. When executing the query via pass-through, we can leave it unquoted, because the remote system is handling the parsing of the code.

```plsql
VARCHAR2(100);  
c    INTEGER;  
nr   INTEGER;  
BEGIN  
  select 'Date in Native Format='||"date1" into val  
  from datetest@findemolink;  
  DBMS_OUTPUT.PUT_LINE(val);  
  select 'Today in Native Format='||sysdate into  
  val from dual;  
  DBMS_OUTPUT.PUT_LINE(val);  

  c := DBMS_HS_PASSTHROUGH.OPEN_CURSOR@findemolink;  
  DBMS_HS_PASSTHROUGH.PARSE@findemolink(c,  
    'select ''Date in YYYYMMDD=''+convert(varchar(20), date1, 112) from datetest');  
  LOOP  
    nr := DBMS_HS_PASSTHROUGH.FETCH_ROW@findemolink(c);  
    EXIT WHEN nr = 0;  
    DBMS_HS_PASSTHROUGH.GET_VALUE@findemolink(c, 1,  
      val);  
    DBMS_OUTPUT.PUT_LINE(val);  
  END LOOP;  
  DBMS_HS_PASSTHROUGH.CLOSE_CURSOR@findemolink(c);  
  c := DBMS_HS_PASSTHROUGH.OPEN_CURSOR@findemolink;  
  DBMS_HS_PASSTHROUGH.PARSE@findemolink(c,  
    'select ''Today in YYYYMMDD=''+convert(varchar(20), getdate(), 112)');  
  nr := DBMS_HS_PASSTHROUGH.FETCH_ROW@findemolink(c);  
  DBMS_HS_PASSTHROUGH.GET_VALUE@findemolink(c, 1,  
    val);  
  DBMS_OUTPUT.PUT_LINE(val);  
  c := DBMS_HS_PASSTHROUGH.OPEN_CURSOR@findemolink;  
END;  
/
```

This PL/SQL Block gives us the following output (note that the original record was inserted on the 2nd, and the second set of queries were run on the 7th, hence the difference in the values of GETDATE()):

```
Date in Native Format=02-JAN-12  
Today in Native Format=07-JAN-12  
Date in YYYYMMDD=20120102  
Today in YYYYMMDD=20120107
```

Statements can also be executed via the EXECUTE_IMMEDIATE program – for things such as inserts, deletes or updates.

The value of the “date1” column in the datetest table is January 2, 2012. For this test we will alter the value of that date by setting it to today’s date (January 8, 2012). To do that, we can use the EXECUTE_IMMEDIATE procedure of the DBMS_HS_PASSTHROUGH package.

```
SQL> select * from datetest@findemolink;  
```

```
date1  
---------  
02-JAN-12
```

```
SQL> DECLARE  
2    c    INTEGER;  
3  BEGIN  
4    c := DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@findemolink('update datetest set datel=getdate()');  
5    DBMS_OUTPUT.PUT_LINE('Rows updated='||c);  
6  END;  
7  /
```

Rows updated=1

PL/SQL procedure successfully completed.

```
SQL> select * from datetest@findemolink;  
```

```
date1  
---------  
08-JAN-12
```

Stored procedures can be similarly called the same way as long as only very basic options are used. Full functionality for remote executed PL/SQL procedures is not supported under Generic Connectivity.

Creating a stored procedure on SQL Server to simply increase a date by 1 day would look like this:

```
create procedure test1  
as  
  begin  
    update FinanceDemo .dbo. datetest set date1 = date1+ 1;  
  end;  
```

Then calling it from Oracle would result in the following:

```
SQL> select * from datetest@findemolink;  
```

```
date1  
---------  
08-JAN-12
```

```
SQL> DECLARE  
2    c    INTEGER;  
3  BEGIN  
4    c := DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@findemolink('update datetest set datel=getdate()');  
5    DBMS_OUTPUT.PUT_LINE('Rows updated='||c);  
6  END;  
7  /
```

Rows updated=1

PL/SQL procedure successfully completed.

```
SQL> select * from datetest@findemolink;  
```

```
date1  
---------  
08-JAN-12
```

```
SQL> DECLARE  
2    c    INTEGER;  
3  BEGIN  
4    c := DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@findemolink('update datetest set datel=getdate()');  
5    DBMS_OUTPUT.PUT_LINE('Rows updated='||c);  
6  END;  
7  /
```

Rows updated=1

PL/SQL procedure successfully completed.

```
SQL> select * from datetest@findemolink;  
```

```
date1  
---------  
08-JAN-12
```

```
SQL> DECLARE  
2    c    INTEGER;  
3  BEGIN  
4    c := DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@findemolink('update datetest set datel=getdate()');  
5    DBMS_OUTPUT.PUT_LINE('Rows updated='||c);  
6  END;  
7  /
```

Rows updated=1

PL/SQL procedure successfully completed.

```
SQL> select * from datetest@findemolink;  
```

```
date1  
---------  
08-JAN-12
```
findemolink('exec dbo.test1');
5 END;
6 /
PL/SQL procedure successfully completed.

SQL> select * from datetest@findemolink;

datel
---------
09-JAN-12

Be sure to commit your transactions on the Oracle side once complete, otherwise the lock will be held on the non-Oracle side until the transaction is committed. This is a particular thing to note when dealing with SQL Server from an Oracle database.

Oracle maintains a very optimistic view of concurrency. If you try to query a row that has a pending update against it, Oracle will return the pre-update version of the row. SQL server operates the reverse way, taking a pessimistic view on concurrency. If you try to query a row with a pending update, by default SQL Server will not return the value to you but will wait until the update is either committed or rolled back before returning the result set.

With this in mind, if you are updating across a database link be sure to not unintentionally hold locks in the remote system by not finishing your transaction on the Oracle side.

Conclusion

I have only touched the surface of what Oracle Heterogeneous Services can do in this article. Being able to connect to and manipulate data on a non-Oracle system in real time can have huge benefits in your IT infrastructure should you come across the need to make these systems talk to each other. The details and examples above are only the basic beginnings of what Oracle offers in this area.

Disclaimer

Since this is a presentation aimed at the basic introductory level, the setup and configurations used in this article have used only the very basic configuration options as covered in the earliest Oracle Heterogeneous Services documentation, as well as only briefly touching on the setup of database links and some of the other functionality that can be covered in this kind of environment. There have been many changes and improvements over the years to what Oracle offers with Heterogeneous Services and the reader should research those further as needed.

References

- Oracle 10g Heterogeneous Connectivity Administrator’s Guide, Publication B14232-01, June 2005
- Oracle 11g Heterogeneous Connectivity User’s Guide, Publication E11050-01, July 2009
- Oracle 11g Database Gateway Installation and Configuration Guide for Microsoft Windows, Publication E12061-06, September 2012
- Oracle 11g Database Administrator’s Guide, Publication E25494-02, December 2011
- Oracle 11g Database Gateway for ODBC User’s Guide, Publication E12070-03, January 2012

Further Reading


Steve Wales is a Senior Technical Consultant with AddOns, Inc., where he spends his days supporting both Oracle and SQL Server databases across a wide variety of versions and Operating Systems as well as dabbling in Unix system administration and supporting a large ERP system for several customers. An Oracle Certified Professional and Microsoft Certified IT Professional for SQL Server DBA, he has been working with Oracle for 13 years and SQL Server for 5 years and lives in Salt Lake City, Utah with his wife and 4 children. He is looking forward to moving back to the Denver area in the Summer.

Bring Your Horses & Guests
Franktown, CO $640,000
6.88 Acres • In-Ground Pool • Huge Shop
- 4282 Square Feet
- Fully Finished Basement
- Wet Bar, Decks
- Apartment Above Shop
- Bonus Room Above Garage
- Two Fireplaces + WB Stove
- Pasture With Loafing Shed

Rare Close-In Luxury Horse Property
Franktown, CO $925,000
4.5 Acres • Heated Barn • Dressage Arena
- 6339 Square Feet
- Pikes to Longs Peak Views
- Marble & Granite Thruout
- Burnished Cherry Cabinetry
- Full Wing Master Suite
- Two Double Sided Fireplaces

Pat Van Buskirk • Coldwell Banker • Parker Office
(303) 243-0737 Cell (303) 841-5263 Office
www.patvanbuskirk.com pat@patvanbuskirk.com
Focused on Delivering the **right** Solution for the **right** Project

Arisant focuses on understanding business requirements to ensure the most effective implementation of the right solution. Our strategy is aimed at providing an honest and expert brand of consulting services for both the Private and Public Sectors. Contact us today to discuss any of the following services, and more.

- Consulting Services
  - Architecture
  - Analysis
  - Design
  - Implementation
  - Project Management
- Staffing/Support
  - Managed Services
  - Managed Hosting Services
  - Oracle Software Sales
  - Oracle Hardware Sales

---

**Quarterly Education Workshops**

**May 24, 2012 • Oracle/Sun Office**

Join us for our next Quarterly Education Workshop in May for presentations at the Oracle/Sun Office in Broomfield, CO. RMOUG hosts quarterly workshops in May, August and November of each year with the fourth and largest educational event being Training Days in February. Learn about the newest technologies, gain more insight into Oracle techniques and enjoy the camaraderie of meeting with other Oracle professionals.

If you or your organization are interested in partnering with RMOUG to host an upcoming meeting, or to submit an abstract for presentation, please contact

Carolyn Fryc, Programs Director at ProgramsDir@rmoug.org

**Watch RMOUG’s Web Page for May Training Topics**

[www.rmoug.org](http://www.rmoug.org)
I like to joke that I have been involved with Primavera for as long as I have been married, and it is true. In early 2001 I was living in Pasadena, completing a software project, an estimating system, for a client. At the time I had heard of this other project at the same client that was in trouble. They were trying to integrate JD Edwards World with Primavera Expedition 7.0, and it was not going well. The client had been asking that I step in to help, but I was quite happy with my on-budget, on-schedule project, and did not want to join the Titanic, if you know what I mean. But they kept asking, and the estimating project was wrapping up, so I told them that I could start after my wedding.

My lovely wife Heather and I were married on June 30, 2001, and enjoyed a wonderful vacation on Kauai'i. When I returned I started on a project that would change my career path quite dramatically. Up until then I was an ex-physicist turned software developer, creating basically whatever the next client asked for. I had only heard of ERP systems tangentially, when support staff at my college bemoaned this thing called PeopleSoft that was being rolled out. I had heard of project management software, but again, only tangentially from project managers at the Jet Propulsion Laboratory where I worked on satellite projects. But I did not really know anything about these systems. And in retrospect I did not know squat about databases!

Well that project was a thorough learning experience. After a few months the original vendor was let go and I and my team were given responsibility for the full project. The prior vendor had been using a tool that looked absolutely lovely, and appeared to make the integration simple, but instead offered an inflexible, graphical tool that could not be debugged, nor tested, and was stored as a pure binary. Version control was not even on their radar. And there was a clear separation between the consultants, who had little technical experience, and the developers, who had some technical experience but no domain knowledge.

After a few harrowing months with a steep learning curve, we pulled it off. The integrations were put into an actual programming language, we implemented source control, abstracted common functionality, and made a set of integrations that still run to this very day, though we have, of course, updated things as the client moved to OneWorld and EnterpriseOne, from an AS400 to SQL server, and through many versions of Expedition, now called Primavera Contract Manager.

And out of that project sprang my current career. I am now quite familiar with JDE and PeopleSoft, as well as Oracle EBS, Maximo, Deltek, Timberline, Lawson, and many other ERP/accounting/estimating systems. My team has integrated all of these with the Primavera products, which includes Primavera P6 for scheduling and resource management, and Primavera Contract Management. In fact, just this past year, in August 2012, I published my first book - and it is on Primavera P6. Now that was a learning experience, both in how to write a book, and digging into parts of P6 that I did not know as well, but needed to master in order to write about.

How Did I Get Here?

I was born in Louisiana and raised in the small town of Clinton. This is not the Cajun country part that most people are familiar with, but rather more like Mississippi. We raised cows on family land that went back generations. We hunted in the forested hills. We rode horses.

But I tended to prefer spending my time reading books, and as an early teen I traded in my horse for an Atari 800 computer. I remember the night before getting the computer, I had gone to Baton Rouge to stay with my grandmother, and I spent the night writing BASIC code into a spiral notebook for a game I wanted to write as soon as I got the Atari. Since then I have greatly enjoyed coding, and find it relaxing and very rewarding.

I studied physics at LSU, where I worked in a group studying cosmic rays and nuclear spallation. There I learned FORTRAN on a VAX, and even did some PDP-11 work. Most of that was data analysis, which I really enjoyed. After LSU I went to graduate school at the California
Institute of Technology in Pasadena, CA. There, I worked in the same group that launched Voyager, which is still working to this day, sending tiny bits of data from the edge of interstellar space.

There I was introduced to the C language and the Unix family of operating systems. I got to C++, Linux, and all that good core CS stuff, while also creating particle detectors and writing a thesis on satellite data from solar energetic particles.

After I graduated in 1998, I flailed around a bit before realizing that my side-hobby of writing software could actually be a career! It had simply not occurred to me that since I tended to spend an inordinate amount of time writing code and reading about coding that I ought to make that a career.

And so I joined AniWorld, which at that time consisted of my friend Justin Cohen and his plans to grow a company creating custom business software. At that time if your company had two Microsoft Certified Professionals, you automatically qualified as a Microsoft Partner, and were allowed a full MSDN subscription, which gave you most of the basic Microsoft products such as Visual Studio. And that is how I wound up working on the estimating system prior to my introduction to Primavera.

In 2002 Heather and I had our first darling boy, Sean. Since neither of us were from California, we decided to move elsewhere, and chose Colorado as our new home. We live just outside of Boulder now, and in 2006 had our own native, Finn. Both boys attend the neighborhood school, first and fourth grades, and when I am not traveling, I have the privilege to walk them to school in the mornings.

I am now the manager of the Oracle Practice at Partners Consulting. This company was the company that merged with the company that bought AniWorld in 2006. Follow that? Well, what it means is that I am still working with Primavera, interfacing with ERP systems. Now we offer a number of services oriented around Primavera, including hosting, integration, implementation, support, and consulting.

The Boulder office is growing, with clients around the world keeping us busy every day. There are a few core principles in this group. First, our single goal is to help our clients and keep them happy. Everything we do revolves around this. Sometimes we help clients with integrations; sometime we just listen and give advice.

Second, we are experts. Our team not only knows the products we support, but know them well, at a technical level. We are all computer geeks at heart, and embrace the concepts of software craftsmanship. Since we also write applications, we also know how others write applications, and use this common knowledge to master the various APIs and databases that drive the products we support. Third, we are involved. I am the Chairman of the Oracle Primavera SIG, and host meetings and webinars for our 600 and growing membership. We stay involved and informed about what new products Oracle is working on, and recently established a practice for supporting the new Unifier product, which is set to replace Contract Manager. I blog about Primavera, participate in discussion groups, and as I mentioned, wrote a book, which I hope to continue updating and improving as the products evolve.

Ok, enough about work. What else do I do? Well, I am involved with Cub Scouts, and the den leader for Finn’s Tiger Den. I love hiking, bird watching, and singing in our church choir. I am also involved in the local Toastmasters group, where I like to speak on a variety of things such as geology and history, or just practice telling a good story. This Christmas I am getting my wife a bow and I’m sure we will all enjoy many hours piercing hay bales in the back yard!
As the newest member of the RMOUG Board of Directors I am excited by all of the possibilities the user group provides. I certainly wasn’t expecting to join the Board and then immediately become the Social Media Director – however, here I am and I’m enjoying every minute, even when keeping track of the social media outlets (LinkedIn, Facebook and Twitter) and events can be a challenge.

I arrived in the Denver/Boulder area by way of Reno, NV, another mountain town, although the Sierra Nevadas, pretty as they may be, are no match for the beauty of the Rockies. And I didn’t come here alone; along with me there is my lovely wife, Shari, our daughter, Gabi and our son, Zachary. All three have adjusted to the move quite gracefully; Shari volunteers at the school in Aurora, Zachary is starting extra-curricular activities as a third grader and Gabi, our fifth grader, is learning the trombone in the school orchestra. Zachary is in Cub Scouts, Gabi is in Girl Scouts; so our schedule during the week can be hectic. Add to that the occasional weekend trips to the mountains and other close by attractions and our calendar can get full in a hurry. It’s nice that Gabi has taken a serious interest in the kitchen as we get exceptional meals, which include recipes she has created, which gives Shari a well-deserved night off.

I’ve been an Oracle DBA for over 23 years at last count but I wasn’t always a DBA. I started my professional career as a Chemist (with a Bachelor’s in Chemistry) and worked my way up the ladder at the manufacturing laboratory where I was employed. Six years after I was hired on, the company bought a new UNIX server (yes, just one) and a shiny, new copy of Oracle 6.0.24, creating a new position of System Administrator/DBA for me to fill. Thrown headlong into the fray, I read everything I could (which wasn’t much back in 1989) and learned from those around me. When the laboratory was closed I decided that being an Oracle DBA was preferable to being a guinea pig for government regulation so I changed career paths and haven’t looked back. I have been fortunate to have provided services for American Airlines/SABRE, the U.S. Postal Service, WorldCom and SiriusXM radio, among others. Currently I support 7-11 Industries, Pearson Education and Nexeo Solutions, a supplier of chemical products and services; so it appears I’ve come, at least indirectly, full circle. I have also been thrust into the Exadata world (through 7-11) so I’ve been busy digging deep into Exadata to learn all that I can. I am also taking on MySQL in support of other clients; in some ways that’s a refresher course as I remember many things from ‘working’ chat room support for Oracle and MySQL users.

Moving here I was excited to be in close proximity to RMOUG, a user group I was familiar with and which provided amazingly talented people from the pantheon of Oracle greatness to contribute to their Training Days events and also counts among its members several Oracle ACEs and an Oracle ACE Director. I had also known of Kellyn Pot’Vin (then Pedersen) through a mutual association with the Oracle-L list; we ‘talked’ at some length about RMOUG and the need for additional Board members. Kellyn, in her own, unique way, ‘convinced’ me to submit the paperwork for consideration on the Board. The rest, as the saying goes, is history.

I write as well, publishing an Oracle-centric blog and contributing monthly articles to databasejournal.com. It was the blog that caught the eye of my original editor at databasejournal.com, Paul Shread, who contacted me to write for their online publication. I am honored to be contributing to such a publication and I believe that I enjoy writing almost as much as I enjoy being a DBA. For those who are interested the blog is here: http://dfitjarrell.wordpress.com/ and a list of articles I’ve written for databasejournal.com is here: http://www.databasejournal.com/article.php/159710/David-Fitzjarrell.htm

It appears that my career has taken a new direction since moving here, and I’m embracing all of the challenges that I am given. The support of this user group is phenomenal and I can only expect to improve through my association with it. I can’t think of any place I’d rather be.
Meet Your Board

Tim Gorman
President & Vendors Director
E-mail: President@rmoug.org

Kelly Pot’vin
Training Days Director
E-mail: TrainingDaysDir@rmoug.org

David Fitzjarrell
Social Media Director
E-mail: TrainingDaysDir@rmoug.org

Kathy Robb
Board Member Emeritus
Arisant, LLC
E-mail: BoardME@rmoug.org

Ron Bich
Secretary & SIGS Director
E-mail: SIGSDir@rmoug.org

Thomas Green
Scholarship Director
E-mail: ScholarshipDir@rmoug.org

Carolyn Fryc
Vice President & Programs Director
E-mail: ProgramsDir@rmoug.org

Art Marshall
Web Director
E-mail: ISDir@rmoug.org

Heidi Kuhn
Administrative Assistant
Voice Mail: (303) 948-1786
Fax: (303) 933-6603
E-mail: Admin@rmoug.org

Pat Van Buskirk
Treasurer & Newsletter Director
E-mail: NewsletterDir@rmoug.org
Semi-Joins and [Anti-Joins]  
Do We Use [NOT] IN, or [NOT] EXISTS?

by Gary J. Propeck, Ph.D., Hotsos Instructor

This past year I attended a conference, where the presenter flashed the following line on the screen:

An in-clause subquery should always be rewritten to use an exists-clause subquery.

When I asked the presenter why he felt he could say this, considering the use of semi-joins and anti-joins, he quoted a well-known Oracle expert, stating that it was so because the Oracle expert said it was so. With a little research it was easy to find the Oracle expert did say it was so, several Oracle versions ago.

These days the Oracle database is so complex that we should not take anything that anyone states without some means of proof. Reproducibility is the key. Without getting into all the processing that semi-joins and anti-joins take one can certainly show the validity of a statement such as this. The purpose of this article is to demonstrate how easy it is, via simple tests to prove or disprove such in the database.

(Google – Roger Schrag/semi-joins/anti-joins. Roger Schrag has several web postings which do an excellent job of dissecting how these “partial” joins function.)

Through a toolset, such as the Hotsos Harness Tools, one can determine if there are any differences in either the execution plan, or the amount of work done, when using either subquery clause syntax.

The Oracle syntax for the in-clause may look like this:

```sql
SELECT <tablea>.*
FROM tablea <tablea>
WHERE <tablea>.column
IN (SELECT other_column
    FROM tableb);
```

Then for the exists-clause syntax would resemble this:

```sql
SELECT <tablea>.*
FROM tablea <tablea>
WHERE EXISTS
    (SELECT null FROM tableb tableb
     WHERE <tableb>.other_column = <tablea>.column);
```

We have two tables with related data:
BIG_TAB
2,294,272 rows 32,889 blocks

SMALL_TAB
500 rows 10 blocks

The tables can be joined by the object_id columns in each table. So formulating two queries to compare:

```sql
ainex1.sql
select count(subobject_name)
from big_tab
where object_id in ( select object_id from small_tab )
```

```sql
ainex2.sql
select count(b.subobject_name)
from big_tab b
where exists ( select null from small_tab s
    where s.object_id = b.object_id )
```

Aliases were used in the second file. This will help distinguish the two statements’ statistics and plans, as the optimizer will use and show those aliases in the trace files and in the explain plans. It will also tell the CBO the two columns come from different tables. All this work was performed on Oracle version 11.2.0.2 (Linux).

Executing each statement in a unit testing environment such as the Hotsos Harness Tools allows us to compare execution statistics and execution plans. As can be seen from the first comparison, the plan cost calculated by the Cost Based Optimizer (CBO) for each statement is the same – 5785.

```
SQL> @bcosts
Enter the workspace name: in_or_exists
```

```
Plan Cost Workspace Scenario
---------- ----------------- ------------------
 5785 IN_OR_EXISTS IN_CLAUSE
 5785 IN_OR_EXISTS EXISTS_CLAUSE
```

Looking at the work performed in each testing scenario through the number of logical I/Os (LIO) again shows identical number of Total LIO (made up of Consistent Gets) for each SQL:

```
SQL> @hlio
Enter the workspace name: in_or_exists
```

```
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Total LIO</th>
<th>Consistent Gets</th>
<th>Buffer Pinned Ct</th>
<th>DB Block Gets</th>
<th>Physical Reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXISTS_CLAUSE</td>
<td>32,668</td>
<td>32,668</td>
<td>0</td>
<td>0</td>
<td>32,654</td>
</tr>
<tr>
<td>IN_CLAUSE</td>
<td>32,668</td>
<td>32,668</td>
<td>0</td>
<td>0</td>
<td>32,654</td>
</tr>
</tbody>
</table>
```

Examining the explain plans for the two SQL statements also one sees the same steps and statistics for both SQL statements.
The Query Transformation Step occurs before the CBO loads the statistics from the data dictionary, and before it begins to calculate costs for the different access and join methods. Later in the trace files, during the evaluation of the join methods, the CBO invokes the semi-join calculations when evaluating the hash join method. In step ID #2 of the explain plans displayed previously, the access predicate is applied and, as can be seen from the plans, the predicates are those from the transformed queries also displayed above.

The Query Transformation Step occurs before the CBO loads the statistics from the data dictionary, and before it begins to calculate costs for the different access and join methods. Later in the trace files, during the evaluation of the join methods, the CBO invokes the semi-join calculations when evaluating the hash join method. In step ID #2 of the explain plans displayed previously, the access predicate is applied and, as can be seen from the plans, the predicates are those from the transformed queries also displayed above.

There are some fundamental requirements to have the CBO invoke the semi-join method.

- Only one table is allowed in the subquery
- The subquery must be correlated with an equality predicate
- The use of a DISTINCT operation (explicitly or implicitly through a set operator such as UNION) voids semi-join use
- Semi-join access paths can be used in queries that contain ORs in the WHERE clause, just as long as the EXISTS or IN is not part of the OR
- The subquery must have no GROUP BY, CONNECT BY or ROWNUM

This raises the next question: What execution plans and statistics are there when we don’t use the semi-join method? For this we can add the no semijoin optimizer hint into each subquery. This will prevent the optimizer from transforming the query by unnesting the subquery and merging it with the main query as we saw previously from the lines extracted from the 10053 CBO trace files.

<table>
<thead>
<tr>
<th>ID</th>
<th>PID</th>
<th>POS</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>11</td>
<td>5785</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>SORT AGGREGATE</td>
<td></td>
<td>1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>*2</td>
<td>1</td>
<td>1</td>
<td>HASH JOIN</td>
<td>BIG_TAB</td>
<td>14877</td>
<td>163647</td>
<td>5785</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>INDEX FAST</td>
<td>SMALL_IDX</td>
<td>500</td>
<td>2000</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>TABLE ACCESS</td>
<td>BIG_TAB</td>
<td>2294272</td>
<td></td>
<td>5762</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "OBJECT_ID"="OBJECT_ID"

SQL> @get_exp_plan_g in_or_exists:exists_clause
Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "S"."OBJECT_ID"="B"."OBJECT_ID"

Final evidence that the processing is the same comes from the 10053 trace files (The CBO trace of the hard parse step). Here we see the same transformed query in 10053 trace files for processing both queries is essentially the same:

Final query after transformations:****** UNPARSED QUERY IS ******
SELECT COUNT("BIG_TAB"."SUBOBJECT_NAME")
"COUNT(SUBOBJECT_NAME)"
FROM "OP"."SMALL_TAB" “S”,"OP"."BIG_TAB" "B"
WHERE “S”."OBJECT_ID”="B"."OBJECT_ID"

kkoqbc: optimizing query block
SEL$5DA710D3 (#1)
::
<call(in-use=1420, alloc=16360), compile(in-use=132480, alloc=166864), execution(in-use=183824, alloc=183868)>

kkoqbc-subheap (create addr=0x84fb38)

QUERY BLOCK TEXT
**************************
select count(b.subobject_name)
from big_tab
where object_id in ( select object_id from small_tab 
where s.object_id = b.object_id )

Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "OBJECT_ID"="OBJECT_ID"

SQL> @get_exp_plan_g in_or_exists:exists_clause
Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "S"."OBJECT_ID"="B"."OBJECT_ID"

Final query after transformations:****** UNPARSED QUERY IS ******
SELECT COUNT("B"."SUBOBJECT_NAME")
"COUNT(B.SUBOBJECT_NAME)"
FROM "OP"."SMALL_TAB" "S","OP"."BIG_TAB" "B"
WHERE "S"."OBJECT_ID”="B"."OBJECT_ID"

kkoqbc: optimizing query block
SEL$5DA710D3 (#1)
::
<call(in-use=1408, alloc=16360), compile(in-use=132480, alloc=166864), execution(in-use=183824, alloc=183868)>

kkoqbc-subheap (create addr=0x84fb38)

QUERY BLOCK TEXT
**************************
select count(b.subobject_name)
from big_tab
where object_id in ( select object_id from small_tab 
where s.object_id = b.object_id )

Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "OBJECT_ID"="OBJECT_ID"

SQL> @get_exp_plan_g in_or_exists:exists_clause
Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "S"."OBJECT_ID"="B"."OBJECT_ID"

Final query after transformations:****** UNPARSED QUERY IS ******
SELECT COUNT("BIG_TAB"."SUBOBJECT_NAME")
"COUNT(SUBOBJECT_NAME)"
FROM "OP"."SMALL_TAB" “S”,"OP"."BIG_TAB" "B"
WHERE “S”."OBJECT_ID”="B"."OBJECT_ID"

kkoqbc: optimizing query block
SEL$5DA710D3 (#1)
::
<call(in-use=1408, alloc=16360), compile(in-use=132480, alloc=166864), execution(in-use=183824, alloc=183868)>

kkoqbc-subheap (create addr=0x84fb38)

QUERY BLOCK TEXT
**************************
select count(b.subobject_name)
from big_tab
where object_id in ( select object_id from small_tab 
where s.object_id = b.object_id )

Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "OBJECT_ID"="OBJECT_ID"

SQL> @get_exp_plan_g in_or_exists:exists_clause
Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "S"."OBJECT_ID"="B"."OBJECT_ID"

Final query after transformations:****** UNPARSED QUERY IS ******
SELECT COUNT("B"."SUBOBJECT_NAME")
"COUNT(B.SUBOBJECT_NAME)"
FROM "OP"."SMALL_TAB" "S","OP"."BIG_TAB" "B"
WHERE "S"."OBJECT_ID”="B"."OBJECT_ID"

kkoqbc: optimizing query block
SEL$5DA710D3 (#1)
::
<call(in-use=1408, alloc=16360), compile(in-use=132480, alloc=166864), execution(in-use=183824, alloc=183868)>

kkoqbc-subheap (create addr=0x84fb38)

QUERY BLOCK TEXT
**************************
select count(b.subobject_name)
from big_tab
where object_id in ( select object_id from small_tab 
where s.object_id = b.object_id )

Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "OBJECT_ID"="OBJECT_ID"

SQL> @get_exp_plan_g in_or_exists:exists_clause
Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "S"."OBJECT_ID"="B"."OBJECT_ID"

Final query after transformations:****** UNPARSED QUERY IS ******
SELECT COUNT("BIG_TAB"."SUBOBJECT_NAME")
"COUNT(SUBOBJECT_NAME)"
FROM "OP"."SMALL_TAB" “S”,"OP"."BIG_TAB" "B"
WHERE “S”."OBJECT_ID”="B"."OBJECT_ID"

kkoqbc: optimizing query block
SEL$5DA710D3 (#1)
::
<call(in-use=1408, alloc=16360), compile(in-use=132480, alloc=166864), execution(in-use=183824, alloc=183868)>

kkoqbc-subheap (create addr=0x84fb38)

QUERY BLOCK TEXT
**************************
select count(b.subobject_name)
from big_tab
where object_id in ( select object_id from small_tab 
where s.object_id = b.object_id )

Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "OBJECT_ID"="OBJECT_ID"

SQL> @get_exp_plan_g in_or_exists:exists_clause
Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "S"."OBJECT_ID"="B"."OBJECT_ID"
Examine the explain plans now shows a divergence. The plan for the statement using the in-clause now shows the subquery being processed as a view. This view (VW_NSQ_1) is then joined to a full table scan of BIG_TAB.

Once again running a unit test using each SQL statement allows us to compare LIO statistics. Of note in these statistics, is the test using the exists-clause subquery with the NO_SEMIJOIN hint does over 2 million LIOs. The processing takes a row from BIG_TAB, and then accesses the SMALL_IDX index. Since there are over 2 million rows that need to be checked that accounts for the over 2 million LIOs.

The Oracle syntax for the not in-clause may look like this:

```
SELECT <tablea>.*
FROM tablea <tablea>
WHERE <tablea>.column
NOT IN (SELECT other_column
         FROM tableb);
```

Then for the not exists-clause the syntax would resemble this:

```
SELECT <table>.*
FROM tablea <table>
WHERE NOT EXISTS
     (SELECT null FROM tableb
      WHERE <table>.other_column = <table>.column);
```

We will use the same two tables with related data:

<table>
<thead>
<tr>
<th>Table</th>
<th>Rows</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIG_TAB</td>
<td>2,294,272</td>
<td>32,889</td>
</tr>
<tr>
<td>SMALL_TAB</td>
<td>500</td>
<td>10</td>
</tr>
</tbody>
</table>

ID | PID | POS | Operation Name | Rows | Bytes | Cost  |
---|-----|-----|----------------|------|-------|-------|
0  | 0   | 1   | SELECT STATEMENT | 1    | 20    | 5823  |
1  | 0   | 1   | SORT AGGREGATE   | 1    | -     | 20    |
+2 | 1   | 1   | HASH JOIN VW_NWO_1 | 16000 | 320000 | 5823  |
3  | 2   | 1   | VIEW SMALL_IDX   | 500  | 6500  | 3     |
4  | 3   | 1   | HASH UNIQUE SMALLIDX | 500  | 2000  | 3     |
5  | 4   | 1   | INDEX FAST FULL SCAN BIG_TAB | 500  | 2000  | 2     |
6  | 2   | 2   | TABLE ACCESS FULL SMALL_IDX | 2294272 | # | 5789  |

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Total LIO</th>
<th>Consistent Gets</th>
<th>Buffer Pinned Ct</th>
<th>DB Block Gets</th>
<th>Physical Reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AINEX1_NO_HINT</td>
<td>32,668</td>
<td>32,668</td>
<td>0</td>
<td>0</td>
<td>32,654</td>
</tr>
<tr>
<td>EXISTS_CLAUSE</td>
<td>32,668</td>
<td>32,668</td>
<td>0</td>
<td>0</td>
<td>32,654</td>
</tr>
<tr>
<td>IN_CLAUSE</td>
<td>32,668</td>
<td>32,668</td>
<td>0</td>
<td>0</td>
<td>32,654</td>
</tr>
<tr>
<td>AINEX2_NO_HINT</td>
<td>2,263,830</td>
<td>2,263,830</td>
<td>0</td>
<td>0</td>
<td>32,654</td>
</tr>
</tbody>
</table>
The tables can be joined by the object_id columns in each table.

So formulating two queries to compare:

```sql
not_ainx1.sql
select count(subobject_name)
from big_tab
where object_id not in ( select object_id
from small_tab )
```

```sql
not_ainx2.sql
select count(b.subobject_name)
from big_tab b
where not exists ( select null from small_tab
s where s.object_id = b.object_id )
```

Executing each statement in a unit testing environment such as the Hotsos Harness Tools allows us to compare execution statistics and execution plans. As can be seen from the first comparison, the plan cost calculated by the Cost Based Optimizer (CBO) for each statement is the same – 5821.

```sql
SQL> @hcosts
Enter the workspace name: not_in_or_exists
```

Comparing the number of LIOs again shows identical numbers.

```sql
SQL> @hlio
Enter the workspace name: not_in_or_exists
```

The explain plans are also once again identical.

```sql
SQL> @get_exp_plan_g not_in_or_exists:not_in
```

10053 CBO trace file. The interesting thing about the transformed query is that it is the same for all for SQL statements reviewed. In our first two examples (the semi-join), the transformed query is processed using the HASH JOIN RIGHT SEMI row source operation. The same transformed query is processed using the HASH JOIN RIGHT ANTI row source operation in the two using NOT IN and NOT EXISTS.

```sql
== Final query after transformations:******* UNPARSED QUERY IS *******
SELECT COUNT("BIG_TAB"."SUBOBJECT_NAME")
"COUNT(SUBOBJECT_NAME)"
FROM "OP"."SMALL_TAB" "S","OP"."BIG_TAB" "B"
WHERE "BIG_TAB"."OBJECT_ID"="SMALL_TAB"."OBJECT_ID"
"COUNT(B.SUBOBJECT_NAME)"
FROM "OP"."SMALL_TAB" "S","OP"."BIG_TAB" "B"
WHERE "S"."OBJECT_ID"="B"."OBJECT_ID"
```

```sql
== Final query after transformations:******* UNPARSED QUERY IS *******
SELECT COUNT("BIG_TAB"."SUBOBJECT_NAME")
"COUNT(SUBOBJECT_NAME)"
FROM "OP"."SMALL_TAB" "S","OP"."BIG_TAB" "B"
WHERE "S"."OBJECT_ID"="B"."OBJECT_ID"
```

```sql
kkoqbc: optimizing query block SEL5DA710D3
(#1) :
```
```sql
kkoqbc-subheap (create addr=0x7ffb38)
**********
```

```sql
QUERY BLOCK TEXT
**************
select count(subobject_name)
from big_tab
where object_id not in ( select object_id
from small_tab )
```

```sql
select count(subobject_name)
from big_tab
where object_id not in ( select object_id
from small_tab )
```

```sql
Predicate Information (identified by operation id):
2 access - "OBJECT_ID"="OBJECT_ID"
SQL> @get_exp_plan_g not_in_or_exists:not_exists
```

```sql
Predicate Information (identified by operation id):
2 access - "OBJECT_ID"="OBJECT_ID"
```
**QUERY BLOCK TEXT**

```sql
select count(b.subobject_name) 
from big_tab b 
where not exists 
  ( select null from small_tab s where 
    s.object_id = b.object_id )
```

To reiterate then, the 4 SQL queries (two semi-joins and two anti-joins) are all transformed into essentially the same query. It is the join processing step where the SEMI and ANTI join methods benefit performance. Where the semi-join stopped processing the second row source when a match was found, and the row was returned from the first row source, the anti-join stopped processing the second row source when a match was found, the row was not returned. The only time a row in the anti-join from the first row source will be returned would be when the second row source is processed and no match is found from the first row source. In both these cases, the benefit of the semi-join and anti-join is the end of the processing when the match is found. Much of the literature mentions that a row is returned from the driving row source for the semi-join when one or more matches are found in the second row source. The point here, is that it never finds a second match, if one or more exists in the second row source, since processing is stopped.

One last issue concerning the anti-join should be discussed. Null Aware.

Here is another example of two queries, one using a not in clause, the other using a not exists clause.

```sql
anotin.sql

select count(*) 
from emp 
where empno not in ( select mgr from emp )

anotex.sql

select count(*) 
from emp eo 
where not exists (select null from emp e where 
  e.mgr = eo.empno)
```

Executing these two queries against the same table gives these results:

```sql
SQL> @anotin
COUNT(*) 
----------
0

SQL> @anotex
COUNT(*) 
----------
8
```

Examining the execution plans for each shows one difference:

```sql
SQL> @get_exp_plan_g antijoin:not_in_na
Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "EMPNO"="MGR"
```

The NA in Step ID #2 in the first explain plan (HASH JOIN ANTI NA) represents the Null Aware situation recognized by the optimizer. This is new in Oracle 11g. When the NOT IN subquery encounters and returns a null value, the clause evaluates to false and no rows are returned. Prior versions of the RDBMS did not process this as an anti-join. Also, one can note that the not exists execution plan includes a filter at Step #4 (4 filter - "E"."MGR" IS NOT NULL). Well, then the logical fix becomes:

```sql
anotin_fix.sql

select count(*) 
from emp 
where empno not in ( select mgr from emp 
  where mgr is not null)
```

```sql
SQL> @anotin_fix
COUNT(*) 
----------
8
```

Now the execution plan becomes:

```sql
SQL> @get_exp_plan_g antijoin:not_in_fix
Predicate Information (identified by operation id):
-----------------------------------------------
2 access - "E"."MGR" IS NOT NULL
```
2 access - "E"."MGR"="EO"."EMPNO"
4 filter - "E"."MGR" IS NOT NULL

In the examples using BIG_TAB and SMALL_TAB, the absence of nulls did not require the predicate to be modified to consider nulls. Again with a simple fix, we see there exists no difference in the processing of the anti-join.

To conclude, as Oracle processes the in-clause subquery and the exists-clause subquery via the semi-join method (returning the same data) the CBO transforms both queries into the same query for processing. The same is said for processing queries via the anti-join method when using the not in-clause or the not exists-clause. Simple testing can show this to be true and allow us to focus our energies on other performance considerations. Oracle’s CBO takes any guessing out of this issue.

Thanks for allowing me to expound. Hope everyone has an incredible time at RMOUG Training Days 2013.

Gary earned a Ph.D. in Physical Organic Chemistry from the University of South Florida in 1984. In 1990, while working at a DOE facility, he began working with the Oracle Database in a chemistry laboratory environment. In 1995, Gary went to work for Oracle Education (now Oracle University) as an instructor. Gary taught the OU curricula for Database Administration, Tools (Forms, Reports, etc.), Oracle Application Server, and Languages (SQL, PL/SQL, and Java). During that time, Gary made numerous presentations at events such as Open World and RMOUG Training Days. In 2002, Gary went to work for Corporate Express as a DBA, gaining more experience working with the Oracle Database. In July 2004, Gary decided to go back to teaching as an independent contractor, teaching much of the same curricula as before, while adding new courses such as OEM and Apex to his repertoire. 2009 brought Gary to Hotsos, as he began teaching as a contractor. In August 2010, Gary started teaching as a Hotsos employee where he teaches the Performance Tuning curriculum.
RMOUG Training Days is the real deal.
Three days of training for $275

RMOUG TRAINING DAYS
February 11-13, 2013 | Denver, Colorado
Colorado Convention Center
www.rmouog.org

Training Days: Tuesday, February 12, 2013 and Wednesday, February 13, 2013
Two full days with more than 140 sessions on the following topics:
• Application Development
• Business Intelligence
• Database Administration
• DBA Deep Dive
• Database Tools of the Trade
• Middleware

Pre-Training Days Events: Monday, February 11, 2013
University Sessions: Six in-depth sessions to choose from.
Master Class: Mastering Oracle Trace Data - Cary Millsap, Method R Corporation
Oracle Technology Network Developer Day: One Day, Hands-on, for Developers and DBAs

Register before January 9, 2013 to take advantage of the Advanced Registration Rate

Are you playing the RMOUG 2013 Training Days Contest: “Do You Know Denver”?
Check it out at http://www.rmouog.org/training/photo-contest/