Using Hyperion Interactive Reporting with the Oracle BI Server

An Oracle White Paper
December 2009

ORACLE

Introduction

The business intelligence toolset originally developed by Brio Technologies has been a very popular product and has its legions of loyal users. Through the years, the product name has changed several times (BrioQuery, Brio Intelligence, Hyperion Intelligence and now Oracle's Hyperion Interactive Reporting) but the product has remained fundamentally the same.

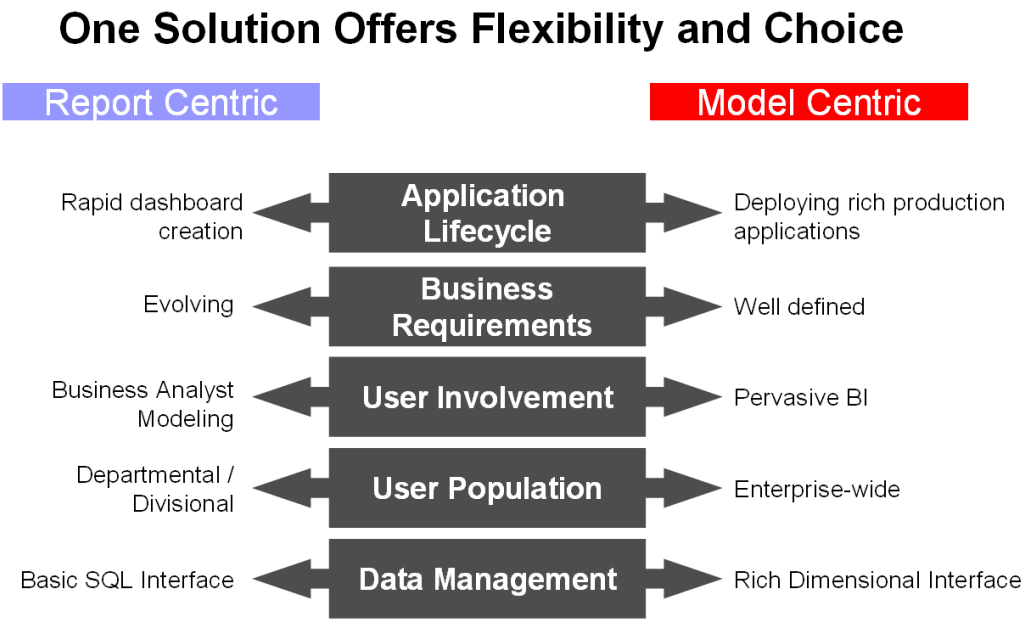
The acquisition of Hyperion by Oracle in June, 2007, however, has left many "Brio" users wondering where to take their business intelligence toolset next. Oracle has a Lifetime Support Policy for Hyperion Interactive Reporting, so it remains a supported product. But where will the next innovations in BI come in the Oracle product line and what can IR users do to further enhance their chosen toolset?

The Interactive Reporting tool uses a report-centric approach to BI that provides a lot of flexibility. Most "Brio" users are used to that level of flexibility and do not want to give it up. However, there are some major limitations with this report-centric approach as well, especially when dealing with complex data environments. IR can deal with these complexities, but they often get surfaced to users in an intimidating manner.

In other words, sometimes IR can be too flexible. This makes it a less than ideal choice for situations where designers need to control what end users can do, or when end users translate "flexible" to mean "too complicated to use". So IR customers also want to know, "How will Oracle address the list of existing IR enhancement requests to provide a better environment for end users, as well as for developers and administrators?"

An answer to both questions lies in the fact that Oracle has integrated Hyperion Interactive Reporting into the overall Oracle Business Intelligence Suite Enterprise Edition Plus (OBIEE Plus) platform. The OBIEE Plus suite is based on a middle-tier query processing engine called the Oracle BI Server. Through this facility, the OBIEE Plus suite can surface new capabilities to IR users by accessing the existing capabilities of the Oracle BI Server rather than adding this functionality directly into the Interactive Reporting code base. This strategy provides a large number of net-new features for IR users without the normally lengthy development cycles required in the past.

The Oracle BI Server is a mid-tier query processor that uses a Model-centric approach to Business Intelligence.



This model-centric approach provides a host of capabilities that IR users have been requesting for quite some time. By using the Oracle BI Server as a "virtual data source" instead of directly connecting to the source databases, those model-centric capabilities can now be leveraged transparently by Interactive Reporting.

So what has been changed in Interactive Reporting? Essentially nothing much ***had*** to be changed. The new features of the Oracle BI Server are surfaced to Interactive Reporting via a standard ODBC driver, so it simply looks like any other ODBC-based data source to IR Studio, the IR Web Client and to the Hyperion Data Access Service. Interactive Reporting only required minor modifications to add a new "Oracle BI Server" data source type for ODBC connections to support these new centrally defined capabilities.

That is one of the major advantages of the Oracle BI Server in OBIEE Plus – you can leverage the new features of this model-centric approach for all users regardless of the front-end tool being used. These advantages apply even when using a report-centric tool like Interactive Reporting. This way, end users can continue to operate in the fashion they are used to working and still be able to access the new features located in the Oracle BI Server that their favorite front-end is calling “under the covers”.

The combination of the model-centric Oracle BI Server and the flexibility of Interactive Reporting provides the best of both approaches. This whitepaper discusses what those new model-centric capabilities are and how to leverage them from within standard Interactive Reporting BQY documents. It will also show how to actually increase overall query performance even though you are adding a new server to your environment. Finally, this whitepaper will provide an outline of a methodology for applying these advantages to existing BQY documents in the easiest fashion possible.

An Overview of the Oracle BI Server

The Oracle Business Intelligence Server acts as a query optimizer and processing engine for any front-end tool that can connect to an ODBC data source and send it SQL statements to be processed. The Oracle BI Server does not move data into a separate data mart, but rather optimizes the query it has been asked to process and then sends the optimized query to the data source or sources that actually hold the data.

The Oracle Business Intelligence Server allows Interactive Reporting users to access a simple set of what appear to be tables and fields in an ODBC database. The reality is that IR users are sending a data request to the Oracle BI Server via its ODBC driver. Under the covers, the Oracle BI Server uses a set of query processing and optimization features that add a host of data enhancement capabilities without the end user needing to be aware of the complexities involved.

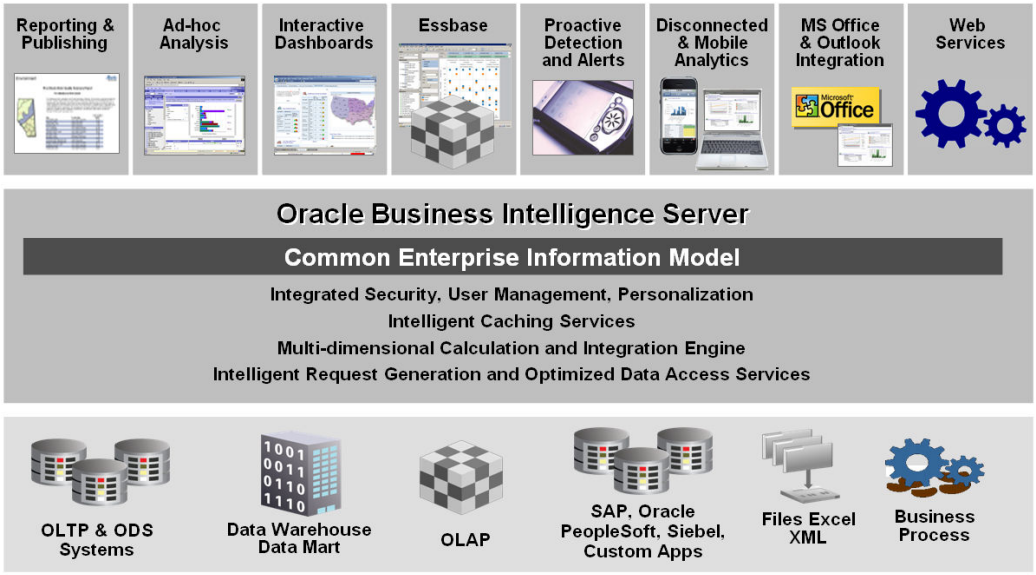


Figure 1 - Oracle BI Server Functional Overview

At the heart of the Oracle BI Server is a metadata layer known as the Common Enterprise Information Model (or CEIM). This model provides the definitions of what physical data sources are to be used, what business model logic applies, and how to present both physical and logical data columns to users in a business-oriented fashion.

Because the Oracle BI Server is based on a centrally managed metadata model, developers can design features one time in a centrally accessed service. This way, queries and reports can leverage the work that has been done once in the Common Enterprise Information Model without having to repeat the design steps over and over in each report. The design work is performed in a GUI tool, so no complex programming is required.

Because this design is centralized outside of the specific Interactive Reporting documents, data enrichment and query optimization tasks can be eliminated from the BQY design process. BQY developers end up with a much simplified process as a result because everything looks like a simple set of tables and columns. The centralized shared design of the Common Enterprise Information Model also supports additional capabilities that normally would require a level of coordination between multiple BQY documents that is simply not possible using IR by itself.

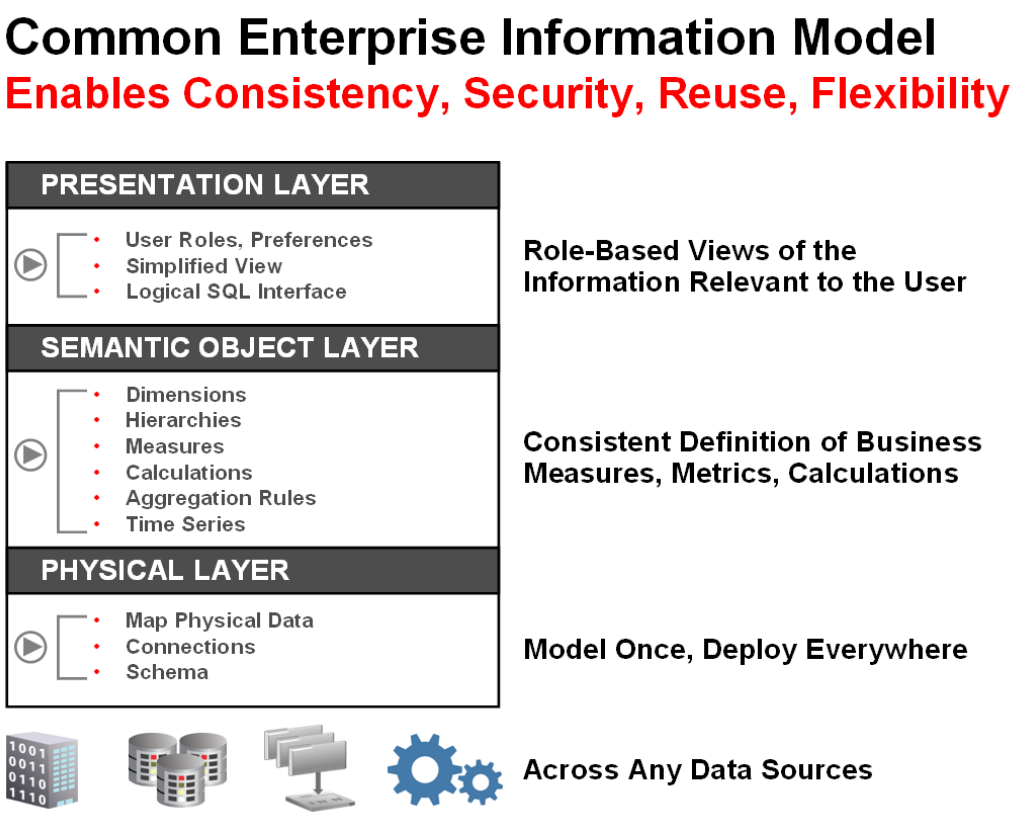


Figure 2 - Functional design layers in the Common Enterprise Information Model

The specific implementation details of the data enrichment and query optimization features of the Oracle Business Intelligence Server are masked from IR users. The IR user only sees the Presentation Layer of the Common Enterprise Information Model. As a result, all the enhanced capabilities are completely transparent to end users. That is why leveraging the Oracle BI Server requires no retraining of IR users. They continue to operate as before yet with a new set of additional features delivered to them.

In general, the specific advantages of this centrally managed approach to information modeling fall into two major areas, business and technical:

ENTERPRISE INFORMATION MODEL BENEFITS	
BUSINESS BENEFITS	**TECHNICAL BENEFITS**
Consistent, Accurate Information	Model Once, Deploy Anywhere
Business-User Self Service	Reduced User Support
Choice of Front-End Tools	Enables Phased BI Standardization
Role-Based, Secure Access	Supports IT Compliance Initiatives
Transparent Information Access	Easier Upgrades & Changes

The specific benefits of using the Oracle BI Server in conjunction with Interactive Reporting fall into three major categories:

- Improving Interactive Reporting Query Performance
- Enhancing Interactive Reporting Security
- Implementation Improvements that make standard Interactive Reporting features either more robust or easier to implement

Using Oracle BI Server to improve IR Query Performance

Many Brio customers expect that insertion of a middle-tier processing engine between a BQY and its data sources would slow queries down. If you only ever had one user accessing one BQY report document using one data source, this might be a reasonable expectation.

In actual practice, BI tools are never used by only one person or for only one report or only for accessing one data source. The opposite is true. You have multiple users accessing multiple reports using a wide variety of databases. So a coordinating query processor helps improve query performance, even for simple queries.

Yet reality is even more complex than that. The multiple databases don't quite coordinate well together. And as the query environment gets more complex, a middle-tier query processor that understands these coordination issues insures that complex queries get designed correctly. That way, users get consistent and correct answers to their questions. So by adding the Oracle BI Server in real-world environments, the performance and data quality improvements can be quite dramatic.

Users become more productive as well. Analysts don't have to waste time trying to figure out how to correctly organize complex queries. IT can get out of the report writing business and concentrate on tasks that only they can do. Managers from different groups can be assured that their numbers are correct and accurate decisions can be made more quickly.

The secret lays in the fundamental differences between a report/document-centric approach (Brio) and a model-centric approach (Oracle BI Server). Every action you take in a regular "Brio" BQY report document happens within the context of that user's instance of that document. There is no cross-document coordination, even between different users of the same document. The BQY document contains a local data cache called the Results section, but one user can not share another user's Results section. So, even if 2 (or 12 or 1200) users execute the same query, it will still be processed again and again and again.

Certainly, database caching will help query performance… if the queries are nearly identical and fetch data from a single source. But what if the queries are different from each other, or if they aggregate data to different levels, or if one is mostly but not entirely a subset of another, or if you really need to process multiple queries across multiple sources? In these cases, single database internal caching doesn't help as much.

Intelligent Shared Caching

The model-centric approach of the Oracle BI Server allows it to provide a centralized multi-user cache that addresses these kinds of real-world situations. The caching implemented by the Oracle BI Server is intelligent, providing support for a wide variety of solutions for real-world optimization.

Automatic

Oracle BI Server caching is a by-product of processing a query. If the physical data source has the "cacheable" property set (which is the default), any queries run against that data source will have the data automatically cached. So, as more people run queries, the system will actually get faster. And since cache loading is based on running queries, you can use a scheduled "after-hours" IR job to pre-fill the cache for ad-hoc users if you choose to. This way, even the first user of the day can take advantage of the performance benefits of the server-based cache.

The cache keeps filling until it reaches the maximum size that you have configured for it. It then uses a least recently used algorithm to control which data remains in the cache and which is replaced.

Shareable and Secured

Probably the most important characteristic of the Oracle BI Server's caching is that it is shareable in a safe manner, even in environments with robust security requirements. When a user performs a query, the query processing engine in the Oracle BI Server first checks to see if the query request can be fulfilled by data that resides in the cache. This cache checking is not limited to what the current user has already queried, so the cached data could have been supplied by a different user's query activity. If all the data needed does reside in the cache, the BI Server then applies any user security restrictions to the data in the cache before sending it to the user. If any of the requested data must be fetched from the database, a new query is formulated to fetch only the missing data. The current user's security restrictions get applied to this new query and the data is merged with what is already found in the cache.

Multi-Source

The Oracle BI Server is designed to assume that the required data resides in multiple sources. Therefore, the data that resides in the cache may have come from more than one source. Users only have to think about which data fields they want, not which data source they came from. The shared caching still works regardless of the back-end physical storage configuration.

Data Subsets

User queries do not have to be identical to be able to take advantage of the shared cache. If one user requests data that is a sub-set of what is in the cache, the Oracle BI Server will subset the data out of the cache automatically and send the requested subset to the user.

As mentioned before, the Oracle BI Server is smart enough to be able to leverage partial subsets as well. If only part of the data needed resides in the cache, the Oracle BI Server can rewrite the user's query to fetch only the missing data and then merge the results with the data subset that already exists in the cache. Therefore, the query processing only hits the back-end database when absolutely necessary and only to fetch the smallest set of data needed to fulfill the user's data request.

Cache-based Re-aggregation

Data aggregations (sum, avg, max, min, etc.) can take a lot of processing power from the database to execute, so these kinds of queries are often the ones that take the most time to complete. The shared cache can be used to re-aggregate data directly out of the cache, even if the cached data is more detailed than is needed. This can eliminate many lengthy round-trips to the database for the most common analytical summary queries used in dashboards and overview reports.

Because the shared caching in the Oracle BI Server is so robust, Interactive Reporting developers may be able to leverage summarized queries to create smaller Results data sets in their BQY files because they no longer need to "query the world" to provide decent drilling performance for end users by staying within a local Results set.

Cache-based Calculations

The Oracle BI Server supports two types of data-enrichment calculations – calculations defined at the server in the Common Enterprise Information Model, and local calculations defined in your BQY query section. Because the query logic in the Oracle BI Server checks the cache first and can subset the data out of the cache, it can also re-calculate formulas based on data in the cache as well.

This means that even ad-hoc users who create a computed item column in their query can still take advantage of the shared cache. The Oracle BI Server supports all the standard calculations and functions that any ODBC data source supplies, so all these calculations take advantage of the shared cache when the base elements of the calculation are all found in the cache.

Automatic Multi-Source Aggregate Navigation

One standard technique for improving query performance for summary operations is to create summary or aggregate tables. The challenge is to create an end-user query environment that takes advantage of these summary tables but only when they apply to the user's current query. This concept of switching to summary tables when applicable is known as Aggregate Navigation.

Unfortunately, setting this up in Interactive Reporting would take a significant effort in custom JavaScript programming in the OnPreProcess event of every BQY document. Not only does this require large amounts of complex code, it is difficult to maintain as well. Any change to the summary table structures would require a rewrite of the JavaScript logic in every BQY. As a result, the benefits of aggregate navigation are not usually available to IR users.

Using the Oracle BI Server as the query processing engine under IR completely eliminates the need to write any custom code to implement Aggregate Navigation. The semantic object layer in the CEIM supports the visual definition of aggregate navigation designs. As you drill into further levels of detail, the system will automatically rewrite the original query so that it points to the aggregate source that is appropriate for the level of summarization requested.

Since the CEIM segregates the physical storage layer from the semantic object layer, aggregate navigation designs are actually independent of the data sources used. This means that you can take advantage of aggregated storage even if it resides in a different data source than the detail data. The multi-source nature of the Oracle BI Server means that, as you drill into further details, the query rewrite mechanism can actually also change which database connection is used to run the query.

For example, you might create an Essbase cube that stores summary data that answers a large number of common end-user questions. If further detail is required, you can drill further and be re-directed automatically to your underlying data warehouse. If you drill past what is stored in your warehouse, the Oracle BI Server can then query the underlying transaction database for the targeted details once a user figures out where in their data environment they need further investigation.

The actual implementation of this kind of aggregate navigation is fairly straight forward. It all takes place in the middle semantic object layer of the Common Enterprise Information Model. A summary of the design tasks breaks down to five basic steps:

1. Drag all the physical fact columns to be used for the aggregate navigation from the physical layer into a logical fact table in the Business Model semantic layer. The figure below illustrates the creation of a Sales Fact logical table from 11 different sources that provide varying levels of aggregation of the same data.

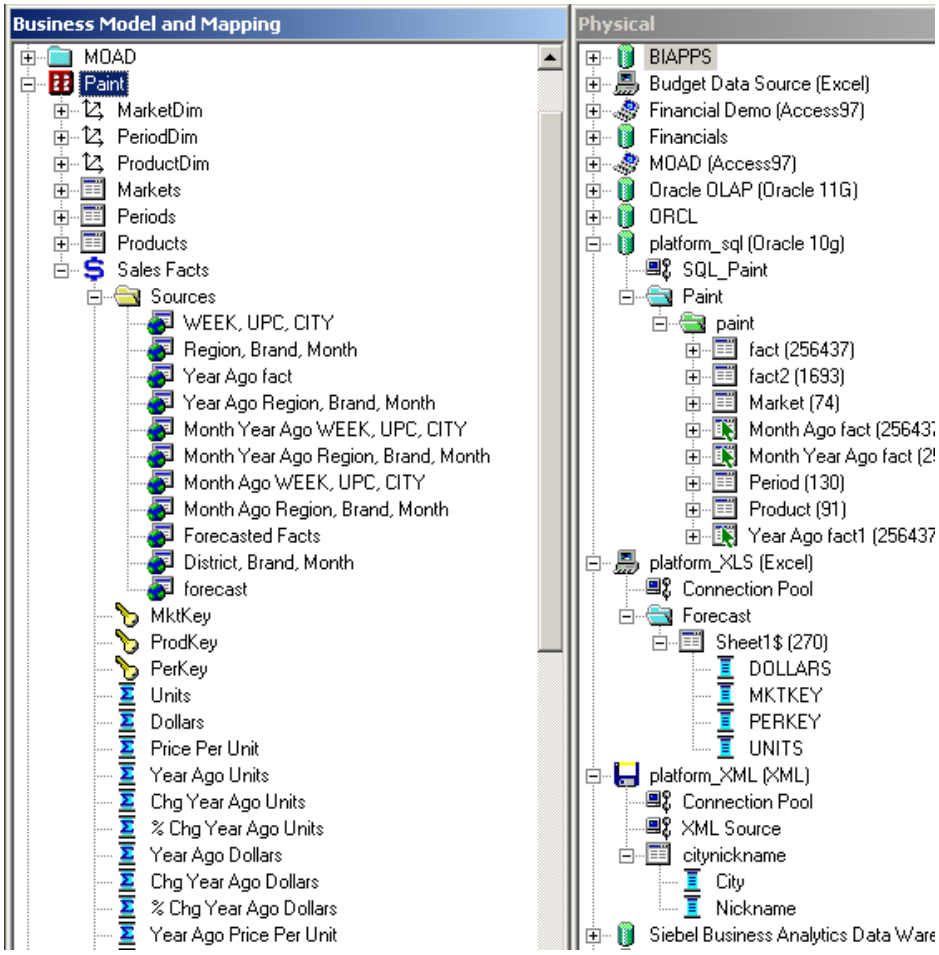


Figure 3 – Logical Fact Table derived from multiple physical sources

2. Repeat this same design process to create the logical dimensions to be used. Drag your dimensional fields into the Business Model semantic layer to create one or more logical dimension tables. The above figure illustrates the creation of three logical dimension tables, named Markets, Periods and Products.

3. Draw a logical business model that illustrates how the logical dimension tables relate to the logical fact table(s).

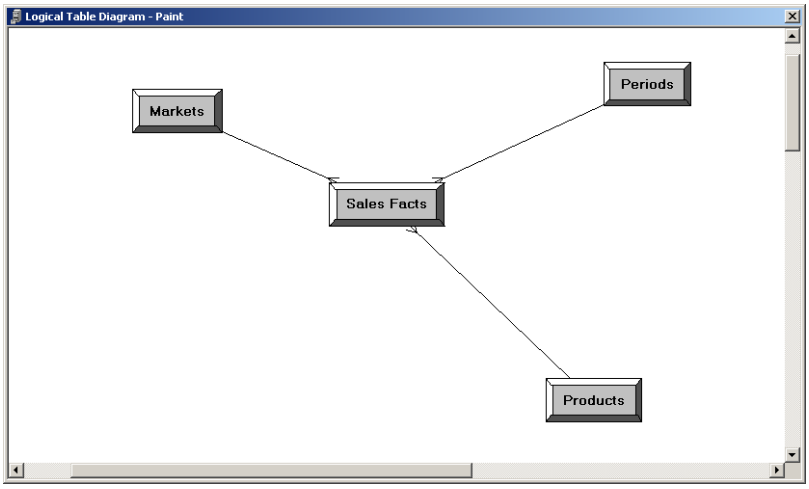


Figure 4 - Logical Business Layer Entity Relationship Diagram

4. Create logical hierarchies that hold the related drill fields from all the sources in a single logical "drill path" hierarchy object. Organize these into the appropriate drill path levels from most summarized to most detailed. For example, you might create a MarketDim hierarchy that organizes a drill path from Region to District to Market. You could also create a ProductDim hierarchy organized from Product Type to Brand to UPC, as shown below.

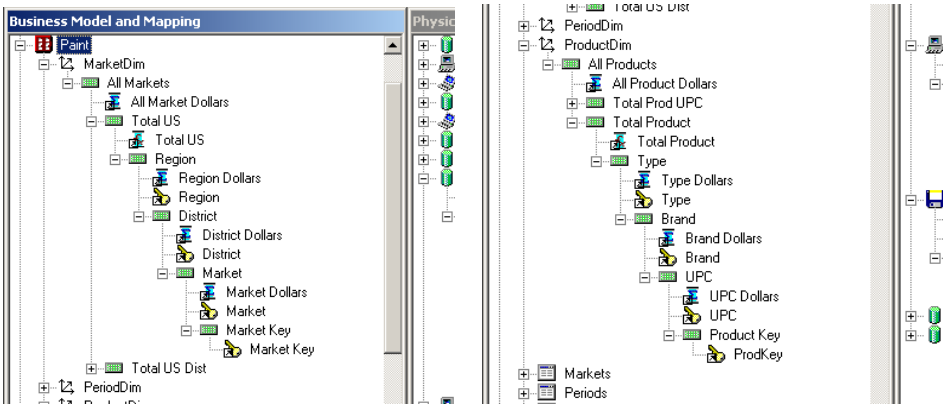


Figure 5 - Drill Path Hierarchies based on the Logical Tables

5. For each aggregate source defined in the logical fact table, right-click, select "Properties" and select the "Content" tab in the dialog box that appears. Here, you will define the hierarchy level(s) that the source covers. The Oracle BI Server will use this information to determine if this specific source contains the information required, or if the query needs to use one of the other sources defined to fulfill a user's specific data request.

   For example, if the logical fact table source named "District, Brand, Month" contains aggregated data down to the Brand level in the ProductDim hierarchy, to the District level in the MarketDim hierarchy, and to the Month level in the PeriodDim hierarchy, you would define its Content properties as follows:

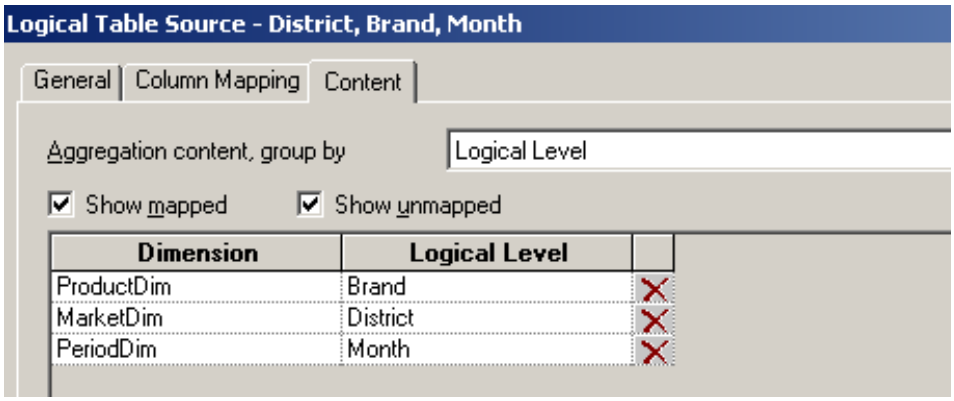


Figure 6 - Defining the level of detail for each aggregate source

Partial Aggregations

Even more complex aggregate navigation scenarios are supported. For example, this aggregate table may only contain data for flat finish paints. The Oracle BI Server supports the concept of query fragmentation (see the later topic below), so you can create these kinds of partial aggregation designs as well.

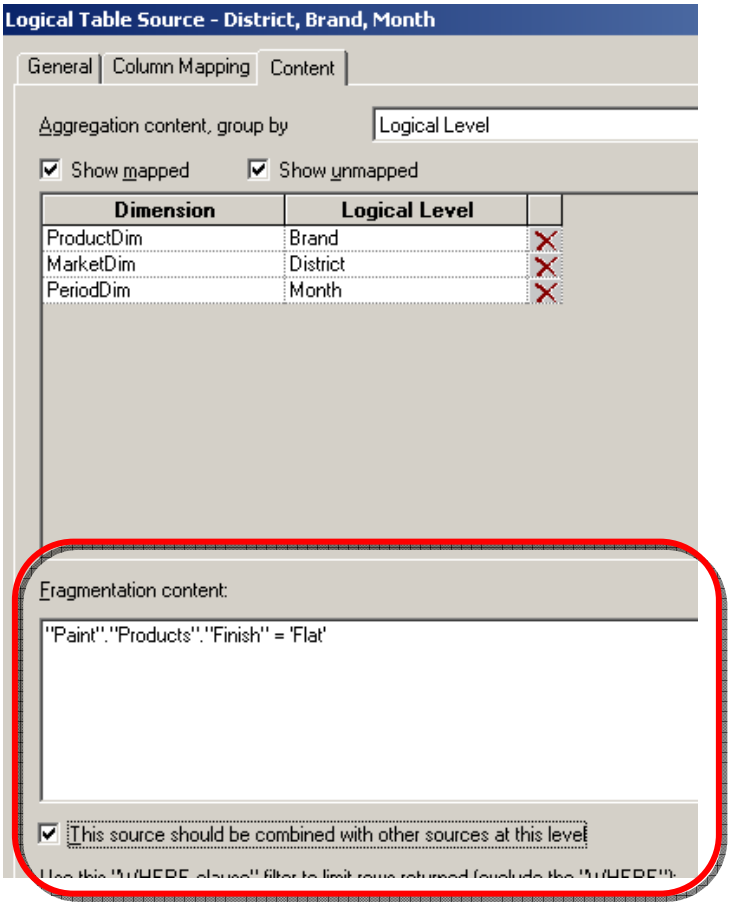


Figure 7 - Defining the data fragment covered by an aggregate source

Once you have finished this logical design, queries will automatically navigate the hierarchy and query the correct data source or sources for the desired level of aggregation. This happens based on the logical tables you created, the fact source contents properties and the relationship diagram that documents how to join the logical tables that lie behind the hierarchy design.

Using the Oracle BI Server Aggregation Wizard

If you don't already have aggregate tables defined, the Oracle BI Server Administration Tool can get you started. It includes a number of helpful wizard-based utilities. One is the Aggregate Persistence Wizard, which is designed to automate the creation of aggregate tables and their mappings into the Oracle BI Server metadata to help boost query performance. This wizard will generate the necessary SQL commands to build an aggregate table based on your responses in the wizard and then map this new aggregate table into the system's metadata.

Refer to Chapter 8, Oracle BI Administration Tool Utilities in the Oracle BI Server Administration Guide for details.

Support for 64-bit Data Access

While most modules in the EPM System 11 platform are available as native 64-bit executables, the BI Service and Data Access Service used by Interactive Reporting are still 32-bit applications. They can still run in a 64-bit OS, but are limited to 32-bits of addressable space. This limits the performance and scalability of an IR-only system.

The Oracle BI Server is available as a native 64-bit executable, and is able to use 64-bit database drivers. This means that queries that are run by the Oracle BI Server can take advantage of a "wider pipe". While the transport of the resulting data set from the Oracle BI Server to the IR report will still be via a 32-bit connection, at least part of the data journey can be made over a higher-performance 64-bit pipeline.

Using Oracle BI Server to enhance IR Query Security

In addition to the query performance gains you will see by using the Oracle BI Server, the model-centric design allows for more control over query security as well. Not only does the Oracle BI Server provide much more robust row and column security than does Interactive Reporting alone, it also supplies a large number of security features that are entirely new for IR users. The Oracle BI Server enables query security features that can:

- Control when a specific user or group is able to query a data source
- Control which logical tables and columns a user can even see
- Automatically "morph" security based on current conditions
- Read in security settings from another application

Session Control

A major US Federal government agency was experiencing a strange problem with their Interactive Reporting system. Each Monday, they had scheduled a series of reports that were needed for weekly management meetings held on Mondays after lunch. But the system started experiencing major slow-downs just when these reports needed to be run. After investigation, it was discovered that a user in a field office would come in early Monday morning and run a series of ad-hoc queries that pulled down 10-15 million rows each. Nothing in the IR system's security set-up could prevent that user from taking the system to its knees.

By adding the Oracle BI Server, the IR system administrators now have the ability to control that user, even though the user is running their queries from the IR Studio desktop client. The Oracle BI Server Session Management allows the administrators to set the days and times when that user can and can not run queries, reserving system resources for the mission critical management reports needed by Monday noon.

Below is an example illustrating how to block queries run against the ERP Projects database by a user account named "capinternal" on Monday mornings, but allow that same user to run queries at other times.

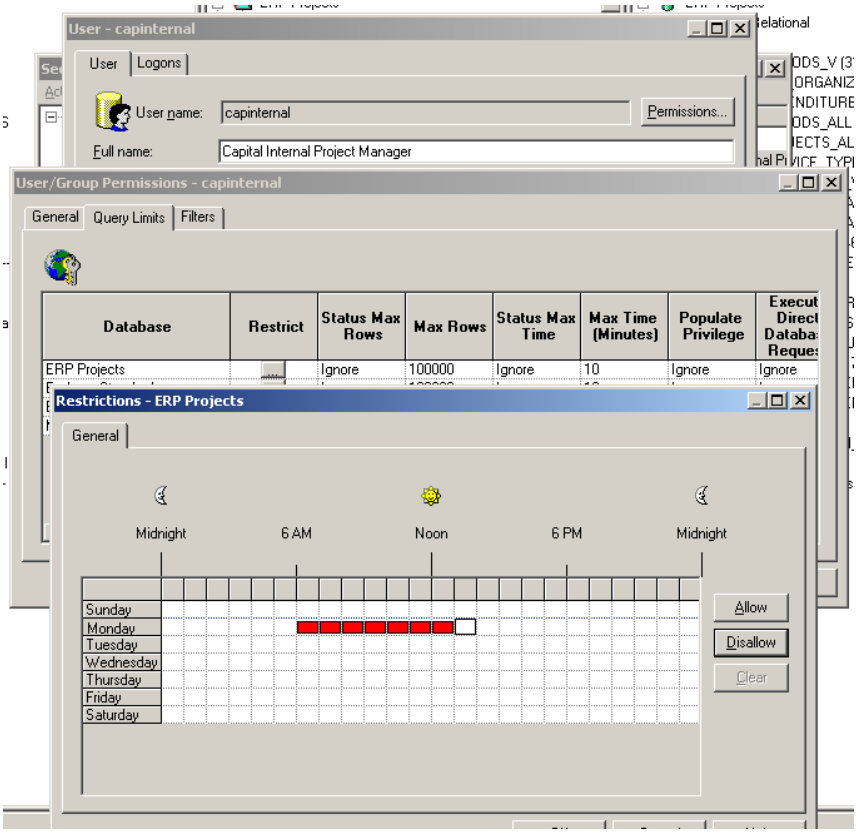


Figure 8 - Defining Query Session Time of Day restrictions

Row & Column Security

Interactive Reporting since v6.6 has provided a rudimentary row-level security feature. While this feature has improved over the years, it only applies to web-based users. The rules can easily be circumvented just by saving the BQY document to the desktop and running it in the IR Studio.

The RLS feature in IR is also not as user-friendly as one might desire. The rules can be complex to define properly, as this feature uses a special set of separate tables containing internal codes to denote different actions. In addition, designers are often confused about the difference between a trigger and a rule, even when using the RLS BQY dashboard to define them. Finally, the IR-based feature requires a separate definition of users and groups that is not coordinated with the overall user authentication system in Hyperion Shared Services.

The RLS feature as defined in the Oracle BI Server is much simpler to understand, define and manage. It is a process that is integrated into the overall design of the Common Enterprise Information Model.

Row-Level Rules

Definition of row-level security rules is a straight-forward process of defining SQL-based filters in the Security Manager module of the Oracle BI Server Administration Tool. You repeat this process for each user or group to whom you want these rules applied. Since you can view and edit multiple rules at once, simple copy-n-paste procedures can be used to duplicate even complex rules. Individual rules can easily be enabled or disabled as needed. The screen below illustrates how easily these rules are defined and managed.

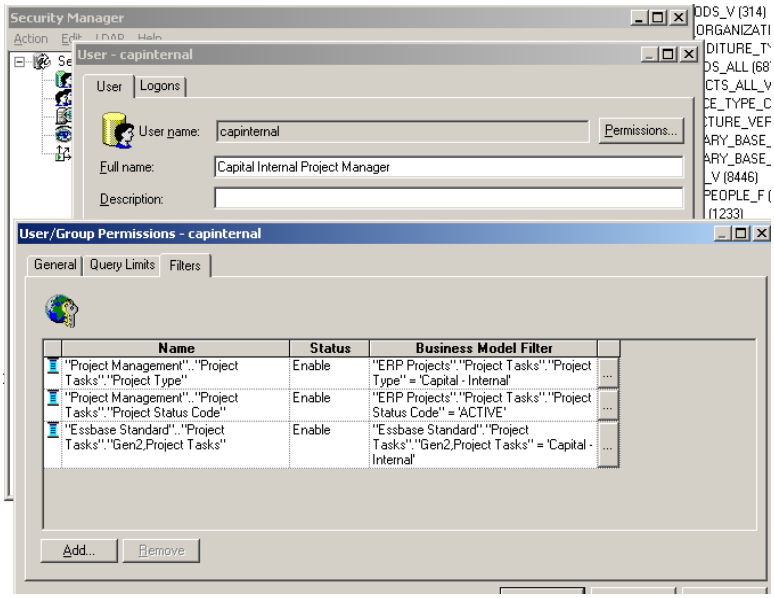


Figure 9 - Defining RLS rules for a user account

Column-Level Rules - Table and Field Visibility

If user security has been designed into the CEIM, then only those elements for which the user has access will be shown to them. Because the specific design implementation in the semantic and physical layers is not surfaced to IR users, this provides an added layer of security that can be applied to Interactive Reporting deployments.

Below is an example of denying visibility to the "Sample FISDM Power User" subject area to any members of the group called "Casual Users". None of the tables and fields in that subject area will be visible to users in that group.

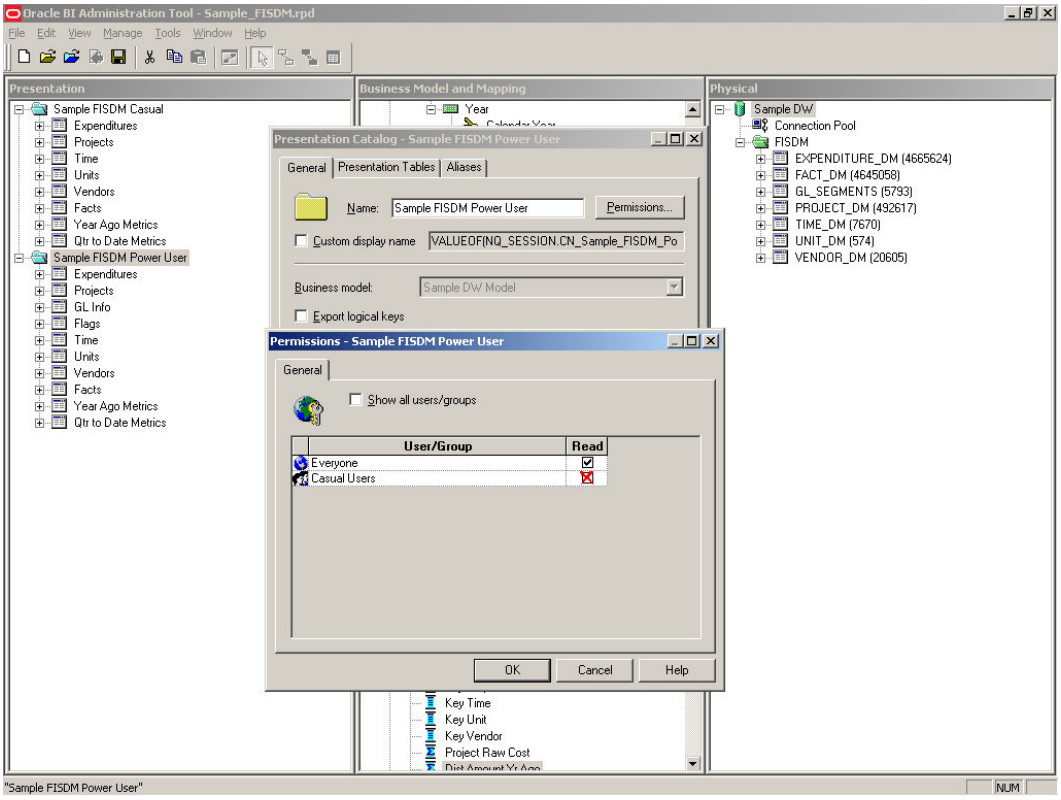


Figure 10 - Controlling Subject Area Visibility

Data Source Access Control

Security can be applied to the physical data connections as well. Specific users may not have rights to access certain data sources, such as the detailed transaction level in a multi-sourced drill-down hierarchy. The CEIM designer can control this very easily by simply denying access to the physical data source itself to that user or group.

Query Governors

The basic feature set of Interactive Reporting includes some rudimentary query governor capabilities. However, these are easily circumvented by simply using the IR Studio tool to allow users the ability to change these settings for themselves. Adding the Oracle BI Server to your architecture provides the ability to move all query controls to a centrally managed environment where end users have no ability to circumvent the rules.

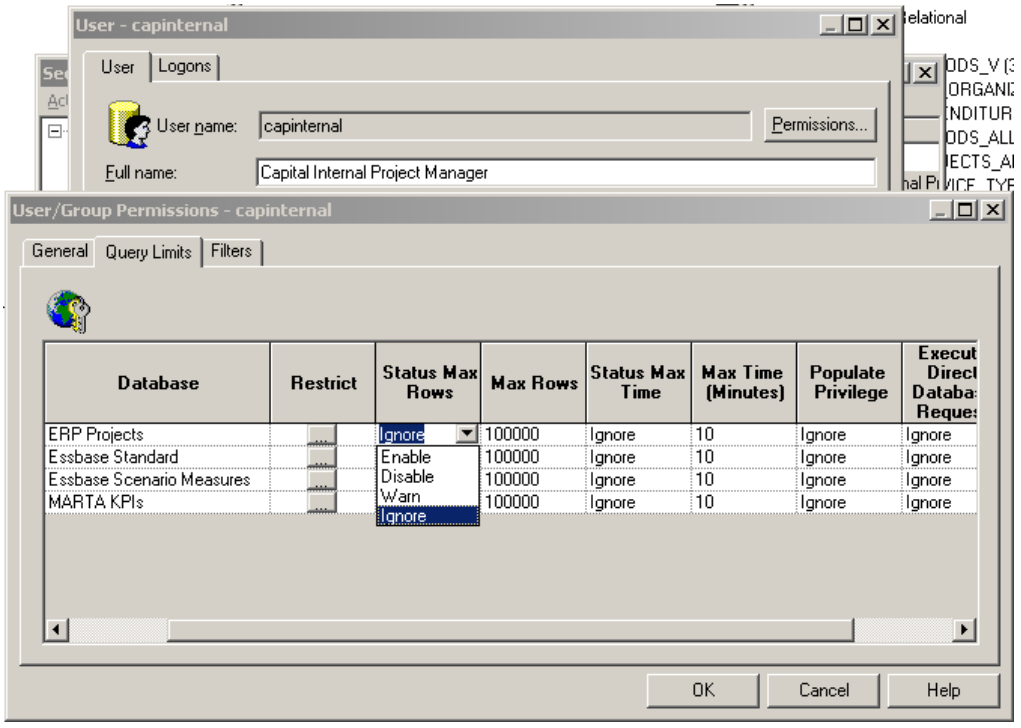


Figure 11 - Robust Query Governors

The above figure illustrates the fact that Oracle BI Server-based query governors are also more robust, more flexible and easier to manage than those offered by Interactive Reporting itself. Query governors can be set to be full fledged maximum settings like in IR, or as mere warnings (something not possible with IR alone). They can be global settings like in IR, or can be targeted to specific users and/or groups, as shown above.

The difference in ease of management goes back to the concept of Interactive Reporting's Report-centric management versus the Oracle BI Server's Model-centric approach. Since all query governors can be moved to the Oracle BI Server, changing the settings becomes a simple matter of changing them in one place, rather than "chasing down" all the BQY and OCE files that have governor settings in them. Centralized management also makes it easier to temporarily disable a governor for special case situations.

Runtime Query Rules

The Oracle BI Server provides support for the definition and use of internal variables. These variables are divided into two major classifications – Repository Variables and Session Variables.

Repository Variables

Repository Variables have values that are set when the BI Server is first started and maintain the same value for all users. They can be either static (hard coded) values or dynamic values that depend on the Oracle BI Server's initial startup environment. A CEIM designer can use Repository variables in expression builders in the Administration tool anywhere you would use a literal or a constant. Repository variables can be used to define rules for defining what is considered "Business Hours", "Closing" or "Quiet" periods, and other standard business definitions that you want under centralized control. By using Repository variables instead of hard-coded values, you enable an easy way to make global design changes in the future, including updates to data source connection strings, default "hidden" data source login credentials, etc.

For example, an application that analyzes when households are watching TV might want to look at a general timeslot from 5pm to 10pm called "Prime Time". The rule could be hard-coded in each calculation as follows:

```
CASE WHEN "Hour" >= 17 AND "Hour" < 23 THEN 'Prime Time' WHEN...
ELSE...END
```

The problem comes when the rule needs to be changed to define Prime Time as 8pm to 11pm. Then you have to search through out all the formulas to update them. If you miss even one, your analysis will be flawed.

Instead, you can define two Repository Variables named "prime_begin" and "prime_end" and implement your formulas as follows:

```
CASE WHEN "Hour" >= VALUEOF("prime_begin")AND "Hour" <
VALUEOF("prime_end") THEN 'Prime Time' WHEN ...  ELSE...END
```

Now if the definition of Prime Time changes, you simply change the value definition of the Repository Variable prime_begin to 20 and of prime_end to 24 in one place, and all your formulas are now updated.

Session Variables

Of even more use for IR developers and users are Session Variables. These variables have their values set when a user logs into the Oracle BI Server to access the Common Enterprise Information Model. Each user receives their own copy of these session variables, and the specific values of each session variable typically vary from user to user.

As with Repository variables, Session variables can be used anywhere in the design of expressions in place of a literal or a constant. This can greatly simplify the creation of rules and actions that should be applied to a large percentage of the total user population, but for which the specifics of the rule needs to be tailored for each individual user.

For example, instead of hard-coding the RLS rules to a specific value for each user or group, a CEIM designer could define one rule for everyone that uses a Session variable as the value to filter on. At run-time, the variable would supply a value or set of values appropriate for the currently logged in user. Thus, definition of one variable-based rule would be all that would be needed to filter each user into seeing only their own data.

How to use Session Variables to implement a User-Specific Global Rule

The Oracle BI Server automatically populates a set of variables known as System Session Variables. Among other items, these system variables include the user's login ID, group membership, display name and email (if defined). But how do custom Session Variables receive values that "morph" depending on who logs in, like System Session Variables do?

The Oracle BI Server provides a design feature known as Initialization Blocks. Both Repository and Session variables can take advantage of Initialization Blocks. Repository Initialization Blocks execute when the Oracle BI Server is first started up. Session Initialization Blocks are executed when a user first logs in to the BI Server.

Initialization Blocks can execute SQL queries that fetch values from a specific data source to determine the actual value of a target Repository or Session variable. Since Initialization Blocks are SQL-based, you could fetch data from a summary table to discover max and min values for defining dynamic query fragmentation rules. You can also use them to fetch filter values from a security table residing in a data source that is separate from your normal query sources.

Setting Dynamic Filters

Session variables support a number of options, including the ability to use other session variables in their definition, support for VPD-based security of values and a feature called row-wise initialization which allows assignment of a list of values to a session variable. Row-wise initialization is very useful for defining dynamic user filters where a user might be allowed to see more than one value in a filtered column.

For example, you might have a security control table named RW_SESSION_VARS that can store multiple rows of data for each User ID. Using another existing session variable named NQ_SESSION.USERID to fetch only those rows that apply to the current user, you might define the initialization block like this:

```
select 'LIST_OF_PRODUCTS', PRODUCT
from RW_SESSION_VARS
where USERID='VALUEOF(NQ_SESSION.USERID)'
```

This initialization block will populate a session variable named "LIST_OF_PRODUCTS" with a colon-separated list of values from the PRODUCT field of the control table. You can then use this variable in a row-level security filter, as shown in the following WHERE clause:

```
where TABLE.PRODUCT = valueof(NQ_SESSION.LIST_OF_PRODUCTS)
```

Defining Sliding Time Period Windows

Initialization blocks can also be used to define a sliding time period to be used for "most recent X periods" types of analysis.

For example, you could define a variable named "current_pa_period" to denote the base time slice for what is considered to be "now" for project analysis sources. The below screen shot illustrates a definition for this type of Session Variable Initialization Block.

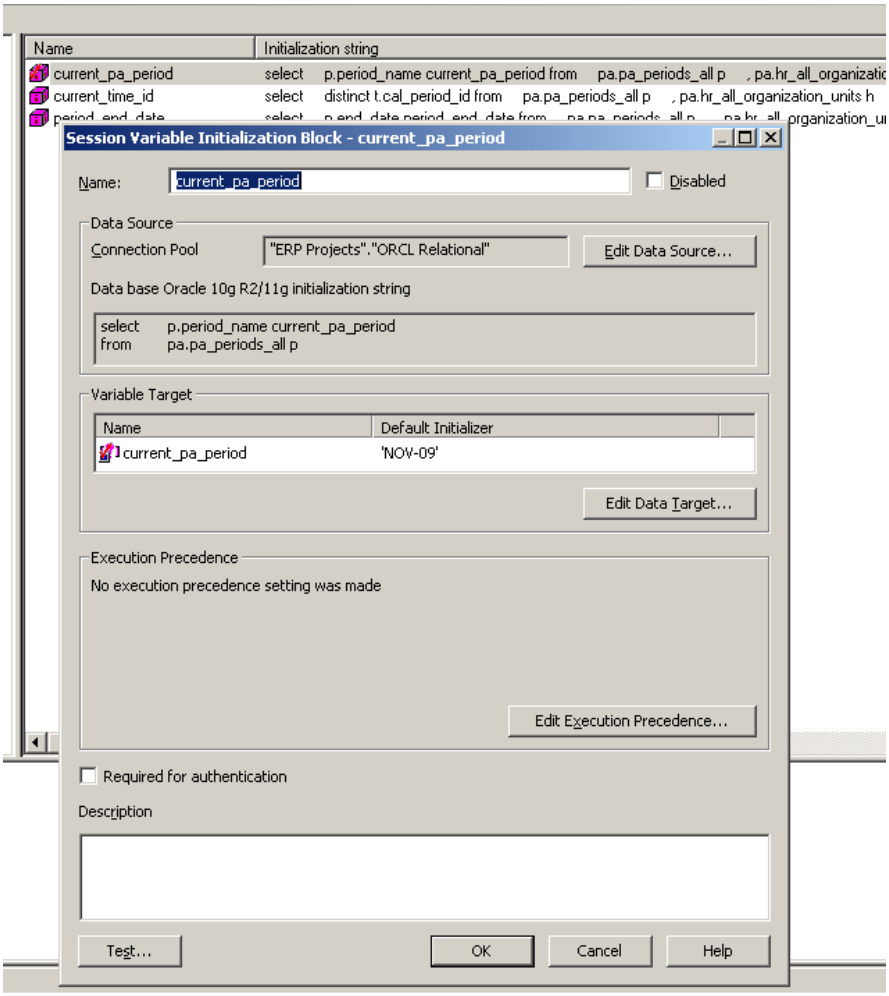


Figure 12 - Setting the value of a sliding time period Session Variable with an Initialization Block

For complete details on using Repository Variables, Session Variables and Initialization Blocks, refer to Chapter 13, Using Variables in the Oracle BI Repository in the Oracle BI Server Administration Guide.

Security Settings based on External Applications

In addition to supporting standard external authentication sources like LDAP and Active Directory, the Oracle BI Server also supports the use of external control tables as a source for user authentication. This capability is typically used to provide user authentication that is based on an external application that stores its authentication information in database tables, such as ERP and CRM systems. This allows one administrator to control the user definitions for both the transaction application that sources the data as well as the business intelligence system that reports on it.

User Session Management

Because all queries are executed via the Oracle BI Server, the oft-requested ability to terminate specific IR end-user queries is now possible. An administrator can log into the Oracle BI Server Administration Tool on the active, on-line system. The administrator can then view and manage active sessions. This includes the ability to terminate specific user query sessions, as shown below, by selecting a request and clicking on the "Kill Request" button. Administrators can also manipulate system variables and clear the shared cache for specific data sources as well.

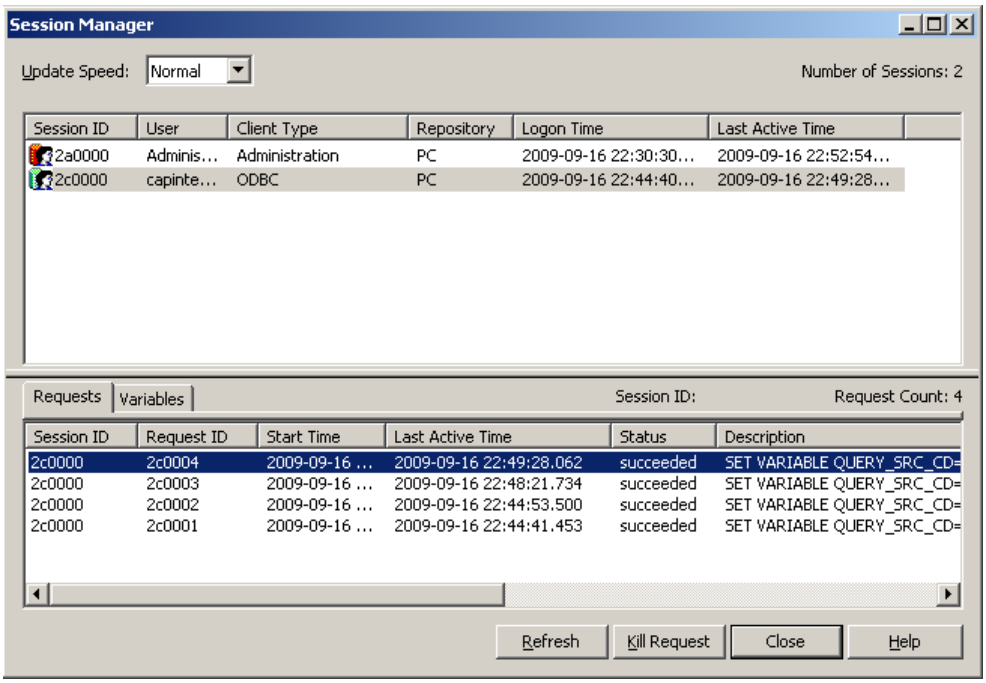


Figure 13 - Active Session Management

Using Oracle BI Server for Implementation Improvements

Some query functionality overlap exists between Interactive Reporting and the Oracle BI Server. However, the centralized way in which the Oracle BI Server implements those features can often be a vast improvement on how you would need to implement that feature using IR by itself. And in many cases, using the Oracle BI Server to implement those features will also eliminate back-door loopholes that users could employ if only IR were used.

Centrally Managed Locked Master Data Models

Database schemas are usually very complex, because the business systems they need to support are complex by necessity. IT organizations need to make sure that this complexity is hidden from users, and that queries and reports are formulated correctly to insure data accuracy.

Using IR alone to accomplish this, you would deploy BQY reporting documents via the EPM Workspace and set the Adaptive Reporting state to nothing higher than Query and Analyze. This setting will not provide end users with the ability to change the joins or add new tables to the query's data model. However, all the user has to do is perform a "Save As" to create a new copy of the BQY file. This then creates a different BQY that the user now owns and they can promote the access privileges to the full "DataModel and Analyze" setting. The once locked data model is now unlocked and available for the user to create some really "creative solutions".

Using the Oracle BI Server changes this dynamic. If the data model definition does not reside in the BQY document itself but in the Common Enterprise Information Model of the Oracle BI Server, end users can be safely given full "DataModel and Analyze" privileges without fear of incorrect query designs.

Moves Maintenance to Centralized Source

The document-centric approach of BQY-based reporting can result in a large system management effort when data model changes arise. At that point, you need to perform an impact analysis and determine what kind of effort you will need to update all your report documents.

With the release of the System 9 and System 11 versions, Interactive Reporting administrators now have a whole collection of tools to help analyze the impact of change and implement automated or semi-automated remediation. But what if you had a system that could avoid most of these kinds of impact to begin with?

That is where the centrally managed data models in the Oracle BI Server can be a huge help. Because the metadata model segregates the Presentation Layer (upon which your BQY reports can be designed) from the Physical and Business Logic Layers, most changes can be implemented with zero impact on the BQY report designs themselves. Only changes that affect the Presentation Layer would require BQY impact analysis.

Centrally Managed Business Names

Businesses often find that different systems store the same kinds of information using different field names. While IR supports the ability to create display names for any field, most users don't use this feature. As a result, reports from different systems can be hard to reconcile with each other.

The metadata in the Oracle BI Server provides an "always on" listing of business-oriented names for each column. End users do not have to remember to use this feature, so it is easy to enforce.

Centrally Managed Standardized Calculation Objects

IR provides a lot of flexibility to users, allowing them to create any kind of calculation they need. Using the Oracle BI Server does not eliminate this flexibility. It does, however, allow CEIM designers to include standard business calculations that enforce standard business rules and definitions. That way, all users know the same calculation can be applied to all reports.

This is especially useful when those business rules change. The CEIM designer can log into the Oracle BI Server Administration Tool, change the calculation definition once in the business logic layer and all reports that use that calculation will automatically receive data based on the new formula the next time the report is run. No BQY updates are required when the calculations are coming from the Oracle BI Server.

Making Multiple Databases look like a Single Data Warehouse

Each Subject Area in the Presentation Layer looks like a simple set of tables and fields to the IR user. Under the covers, that subject area is based on a business model, and that single business model may be based on multiple physical sources, as illustrated below.

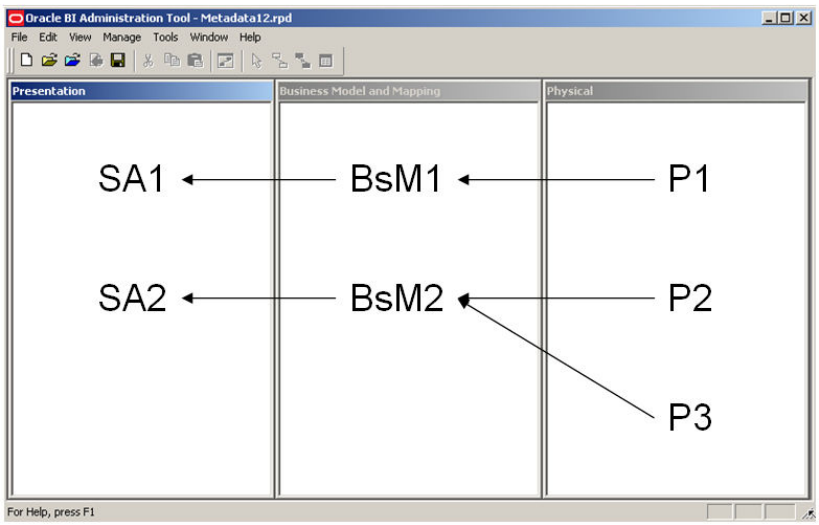


Figure 14 - Designing a Multi-Source Subject Area

Even though multiple physical databases get queried, the user only sees what looks like a single "virtual" data mart.

Eliminating Local Joins with Automatic Multi-Pass SQL

The ability to make multiple sources look like one without actually moving data eliminates the requirement for users to know the details of how to join data from multiple systems. This virtually eliminates the need to use local joins when designing IR reports that pull data from multiple sources.

Under the covers, a query that needs data "joined" from multiple systems will look at the metadata design to determine which table should be the "driver" in the join between sources. The Oracle BI Server will then query the driving data source, create an internal list of unique join values from the result and then use this in a limiting WHERE clause for the query against the next data source.

Sometimes multiple databases are used not simply to join different kinds of data. As discussed in more detail below, sometimes multiple data sources are needed to hold different fragments of the same kind of data. In those cases, support for multi-source query fragmentation allows the Oracle BI Server to federate queries across multiple sources, determining which databases need to be queried for each request.

In this kind of query federation scenario, the controlling WHERE clauses are derived from the Content properties of each source. The Oracle BI Server executes queries against each data source that contains needed fragments, based on what the user requested. Each data fragment is then "stitched together" from the cache into a comprehensive whole sent to the IR report as if the data came from a single table.

Because these Oracle BI Server features eliminate the reasons why users originally needed local joins and local join limits, large multi-source queries no longer need to rely on sufficient desktop resources to successfully complete these kinds of IR reports.

Support for Query Fragmentation

Sometimes so much data exists that the physical storage needs to be split into multiple tables or even multiple databases. A very common storage implementation strategy is to hold more current data (say, current year through current year minus 2) in front-line high-speed storage and older historical data in a separate, less expensive and possibly slower data source. The idea behind this strategy is to provide the best query performance for the data that is most often used.

Of course, the trade-off is that this performance optimization technique makes creating reports more complicated for users if the report needs to go back more than a couple of years. And report designers need to know this ahead of time to create properly designed queries and reports.

This also produces a maintenance challenge in that all such reports may need to be redesigned each year as the sliding window of "the most recent 3 years" changes over time.

Fortunately, the Oracle BI Server knows how to handle these kinds of data storage realities through its support for a capability known as Query Fragmentation. This feature lets you target different portions of your query to different data sources. The end user designs a single, simple query and the Oracle BI Server knows how to fragment that into multiple queries that hit multiple data sources for different subsets of the data.

To set up this capability, you define a logical table in the middle semantic object layer of the CEIM that has multiple table sources defined. You then modify the properties of each table source to define which set of values are covered by each source.

Support for query fragmentation is useful in other data organization scenarios as well. Say, for example, that you are designing reports for a Telecommunications firm. Your report needs to display consolidated information about the firm's entire suite of offerings.

However, this company has two data entry systems for tracking product orders. One system tracks consumer goods and another tracks commercial business offerings. The commercial offerings all are encoded with an entity type code that ranges from 1 to 97. The consumer offerings are encoded with entity numbers above 100. You could tag the fragment of data that comes from the commercial offerings source by setting up its source table as shown in the following figure.

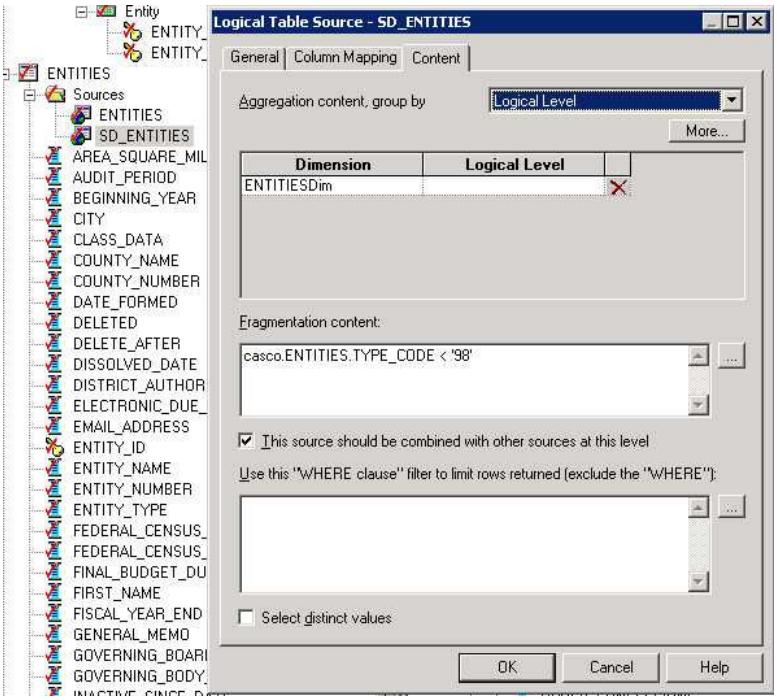


Figure 15 - Defining which data fragment a Logical Table Source covers

In the example shown above, the Entity dimension is coming from two source ENTITIES tables. In addition, the facts are coming from 5 different tables. Using query fragmentation, you can model one logical dimension table using the 2 sources and one logical fact table that is sourced from these 5 underlying tables.

Below is listed an example query that the Oracle BI Server generates to the back-end data sources based on this logical definition. The user built a simple query that merely selected the desired columns. Through the metadata definition in the CEIM, the Oracle BI Server knew the right way to answer this question was much more complex. Note the automatic generation of multiple UNION statements.

```
select D6.c2 as c1,
    sum(D6.c3) as c2
from     ((select case  when D0.c1 is null then 'SD' else D0.c1 end  as
c2,
             D0.c2 as c3
         from              (select T2515.ENTITY_TYPE as c1,
                        T2471.NON_OPEXP_TOT as c2
                   from                         ENTITIES T2515,
                        CO_AIRPORT_ENTERPRISE T2471
                   where  ( T2471.ENTITY_ID = T2515.ENTITY_ID )   ) D0
         union all
         select case  when D0.c1 is null then 'SD' else D0.c1 end  as c2,
             D0.c2 as c3
         from              (select '' as c1,
                        T2636.NON_OPEXP_TOT as c2
                   from                         SD_ENTITIES T2701,
                        SD_AIRPORT_ENTERPRISE T2636
                   where  ( T2636.ENTITY_ID = T2701.ENTITY_ID )    ) D0
         union all
         select case  when D0.c1 is null then 'SD' else D0.c1 end  as c2,
             D0.c2 as c3
         from              (select T2515.ENTITY_TYPE as c1,
                        T2380.NON_OPEXP_NONOP_EXP_TOT as c2
                   from                         ENTITIES T2515,
                        CI_AIRPORT_ENTERPRISE T2380
                   where  ( T2380.ENTITY_ID = T2515.ENTITY_ID )    ) D0)
    ) D6
group by D6.c2 order by c1
```

For situations where the fragmentation is time-based and therefore subject to change each time period as older data gets moved off to near-real-time storage, the use of Repository or Session Variables as mentioned before provides an easy way to implement a self-maintaining system.

Leveraging Intelligent Function Shipping

The Oracle BI Server knows which SQL features are supported by each data source. The Oracle BI Server supports all standard SQL capabilities, even if the data source does not. When a user generates a query request, the Oracle BI Server analyzes the SQL functions requested and determines if the data source is able to fulfill the request natively. If it can, the function is included in the SQL sent to the data source. If not, the Oracle BI Server will fetch the data needed and then perform the needed function in the middle tier.

For example, the RANK function is supported by Oracle relational databases, but not by SQL Server 2000. The Oracle BI Server is smart enough to send the RANK function to an Oracle database, but pull the needed information from SQL Server 2000 systems and perform the ranking from within the cache.

Creating Time-Series Metrics

In designing your business logic layer, you can tag a hierarchy as a Time Dimension. Providing a Time Dimension activates the ability to create time-series metrics. This allows you to create what look like simple fields in the Presentation Layer to perform "single-query" comparisons of data from different time slices.

In the business model layer, for example, you can create a metric called Dist Amount Yr Ago using a simple time-series formula:

```
AGO("Sample DW Model".Facts."Dist Amount", "Sample DW Model"."Time
Dim"."Year", 1)
```

The AGO() function takes three parameters – the data fact column to be compared (the "Dist Amount" fact here), the actual time series dimension level to be used as the basis (Year in this case), and the number of periods ago to look.

Under the covers, this formula will automatically generate multiple queries to pull consolidated data from both the current time slice of the query and the time slice targeted by the AGO() formula. In the example formula shown above, we are looking at data from a year ago. If we used the same formula except with 10 as the number of periods, we would be looking at data from a decade ago.

The Oracle BI Server also provides a similar ToDate() function. These two time series functions can be nested to provide, for example, formulas for finding the "Dist Amount QTD" versus "Dist Amount QTD Yr Ago". You can then combine these features with the ability of the Oracle BI Server to define default aggregation rules for each metric.

These complex multi-pass query definitions get surfaced to IR users as simple fields they can select when building a query. The following screen shot illustrates running such a query from within IR Studio.

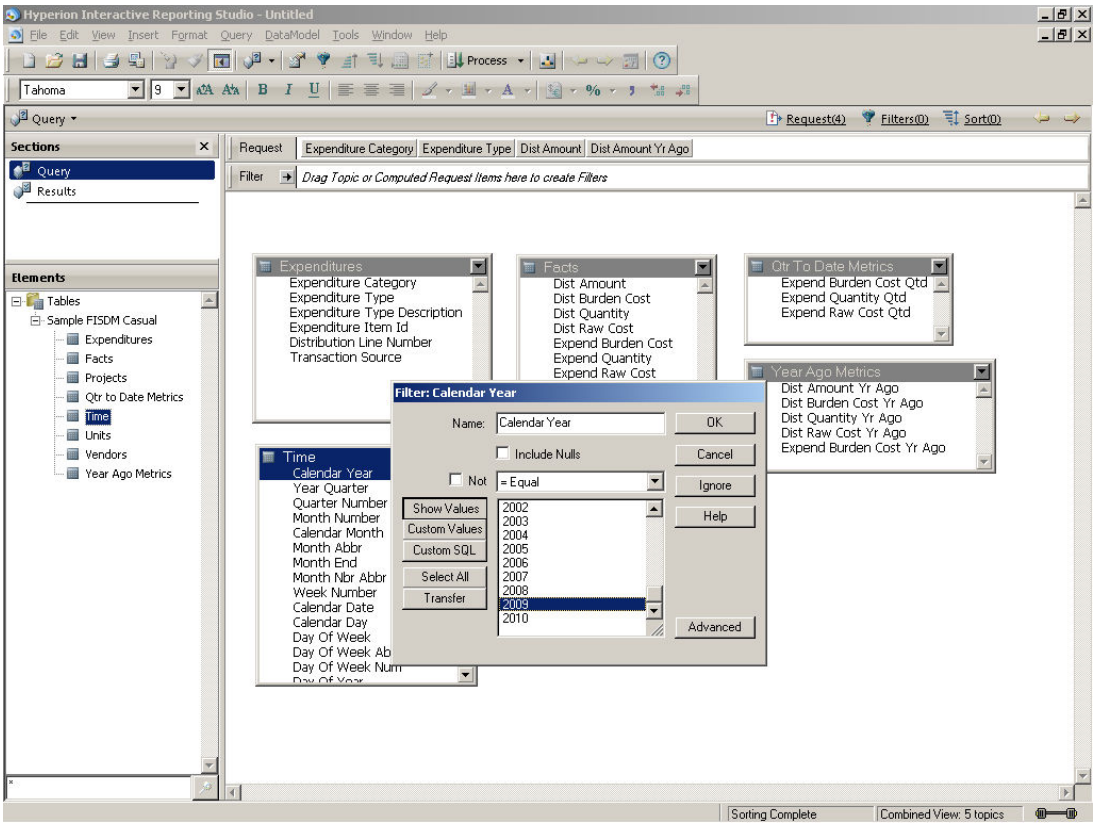
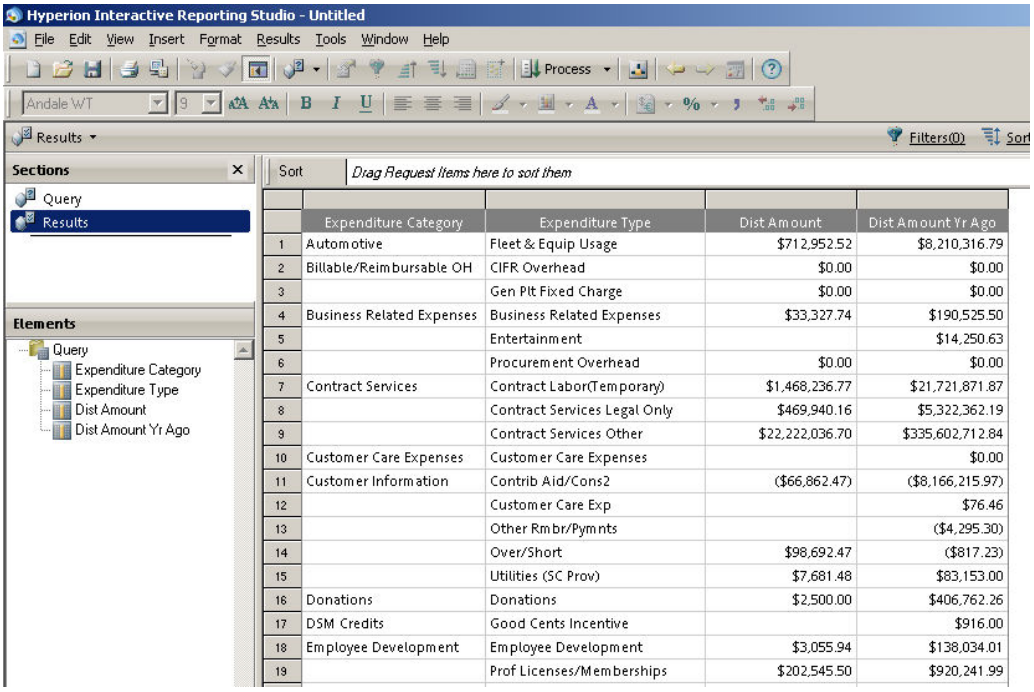


Figure 16 - Running a "simple" Year Ago Query

The above seemingly simple query will fetch the Dist Amount for the selected year (2009 in this case) as well as for the prior year. The user simply includes both the "Dist Amount" and "Dist Amount Yr Ago" fields from the "tables" in the Sample FISDM Casual subject area and creates a normal IR filter. Even the standard "Show Values" function works properly, pulling the related time dimension hierarchy values from the Oracle BI Server transparently.

Notice that even though the data results will require multiple complex queries on the back end, the user only needs to create one simple query in IR. If the data for the different years is stored in different source systems, the Oracle BI Server will automatically federate the queries across the needed sources. The user does not need to create multiple queries for each source and then perform a local join. The user can be totally unaware of the physical implementation of the data sources. And the physical implementation can change without affecting the end user's report design.

The resulting data set appears to the end user as a simple Results section, as shown here.



	Expenditure Category	Expenditure Type	Dist Amount	Dist Amount Yr Ago
1	Automotive	Fleet & Equip Usage	$712,952.52	$8,210,316.79
2	Billable/Reimbursable OH	CIFR Overhead	$0.00	$0.00
3		Gen Plt Fixed Charge	$0.00	$0.00
4	Business Related Expenses	Business Related Expenses	$33,327.74	$190,525.50
5		Entertainment		$14,250.63
6		Procurement Overhead	$0.00	$0.00
7	Contract Services	Contract Labor(Temporary)	$1,468,236.77	$21,721,871.87
8		Contract Services Legal Only	$469,940.16	$5,322,362.19
9		Contract Services Other	$22,222,036.70	$335,602,712.84
10	Customer Care Expenses	Customer Care Expenses		$0.00
11	Customer Information	Contrib Aid/Cons2	($66,862.47)	($8,166,215.97)
12		Customer Care Exp		$76.46
13		Other Rmbr/Pymnts		($4,295.30)
14		Over/Short	$98,692.47	($817.23)
15		Utilities (SC Prov)	$7,681.48	$83,153.00
16	Donations	Donations	$2,500.00	$406,762.26
17	DSM Credits	Good Cents Incentive		$916.00
18	Employee Development	Employee Development	$3,055.94	$138,034.01
19		Prof Licenses/Memberships	$202,545.50	$920,241.99

Figure 17 - Year Ago Query Results

Notice that the definition of an aggregation rule of Sum on both "Dist Amount" and "Dist Amount Yr Ago" provides a simplified data set to the user.

Easing Migration Tasks via Presentation Layer Isolation

One of the most common but difficult tasks is moving what has been created in a development environment into QA and then into production. Because the Oracle BI Server isolates the physical connections from what the user sees to develop reports, this standard lifecycle migration task is greatly simplified.

In environments where the database schemas are the same between development, QA and production, repository variables can be used to define the actual connection strings. To move the entire model from development to QA to production becomes a simple matter of changing the value of the associated repository variable.

In environments where the data sources are different (either different schemas or different technologies), the ability of the business model layer to source logical tables from multiple physical connections becomes the key to lifecycle management. Simply define the logical tables to be connected to all development, QA and production sources, and then disable the physical connections that are not in use in the specific system.

Better Integration with Essbase and Oracle OLAP

Interactive Reporting users tend to want to use IR for all data access, even for non-relational data sources. While this is certainly a possible use of Interactive Reporting, it is not without some challenges. Interactive Reporting assumes that all data sources are like relational sources but OLAP sources such as Essbase and the Oracle OLAP option work in a non-relational way. Adding the Oracle BI Server as an OLAP-aware intermediary can eliminate many of the issues of using a relationally based tool with non-relational sources.

Better Integration with Essbase

The new EssbaseQuery section type introduced with IR v9.3.1 was a vast improvement over previous attempts to integrate Essbase data sources into IR documents. In many cases, these new capabilities are sufficient.

There are a few use cases, however, where the differences between how IR and Essbase "think" make for a complex situation. For example, Essbase provides support for what are known as shared members. Shared members provide alternate ways of looking at the same data. In the standard Essbase Sample:Basic application, users can look at soda sales by flavor (Colas, Root Beer, etc.) or by Diet vs Non-Diet. The sales for Diet Cola appears in both the Colas number and the Diet number. Essbase knows this and when asked to provide a total will automatically know not to count Diet Colas twice.

If you only ever use the EssbaseQuery section to access Essbase data, that section knows to ask Essbase for the proper totals, so you will still not double count Diet Colas. But as soon as you want to marry Essbase data and non-Essbase data into the same report, the EssbaseQuery section can not directly be used for creating consolidated reports. You will need to perform a "Download to Results" on the Essbase data.

The "Download to Results" feature will add columns to the results set when shared members are involved that will mark the duplicate data rows. You will need to use this additional information to design reports that "back out" the duplicate numbers from any reports.

If, however, you use the Oracle BI Server, its native support for Essbase as a part of a multi-source business model provides a much easier solution. You build your IR report from a single subject area that combines Essbase and non-Essbase data, setting the Essbase sources to use the Essbase server to define the aggregation rules for that data source. Then you simple design your IR report as if all the data came from a single relational source.

Better Integration with Oracle OLAP

The Oracle relational database now offers what is known as the OLAP option. Like Essbase, this Oracle database feature creates multi-dimensional cubes for advanced data analysis. Unlike Essbase, the Oracle database surfaces these cubes to end users as relational table views.

At first, this might seem like an ideal environment for using Interactive Reporting on top of analytical cubes. However, because of the way Interactive Reporting operates, the query requirements to obtain correct results make it a bit unwieldy for IR users.

To properly query an Oracle OLAP cube, each query must involve all defined dimensions in the query join. This requirement means you will have to remember to change the Data Model Options setting for Joins to "Use All Joined Topics" for each query that uses Oracle OLAP cubes. In addition, filters need to be set at each level in the dimension. This means that drilling would require adding new fields to the Limits area each time a user wanted to drill. But the Oracle OLAP option uses level-based filters, meaning that the same field is used for all levels in a dimension with a different level defined for each drill filter. Interactive Reporting expects that filter information for different levels are contained in different columns. Finally, since the different levels in an Oracle OLAP dimension are contained in the same column, you would need to create multiple instances of columns to create unique level names for the Results section columns, or have columns with generic dimension names and multiple levels of content. This is not exactly what Interactive Reporting users expect to have to deal with.

The Oracle BI Server, however, can handle all these requirements transparently. When you use the "Export to OBIEE" plug-in in the OLAP Option's Analytic Workspace Manager, it will automatically generate the correct dimensionally-aware objects. The comments from the AWM utility will also document the correct security filter to add to the Oracle BI Server metadata to automatically add all the dimensions to any join. As your Interactive Reporting users drill to further detail, the Oracle BI Server will automatically add each new level-based filter required. So, by using the Oracle BI Server's intimate understanding of the Oracle OLAP Option, Interactive Reporting users can leverage the power of the Oracle cubes without changing the way they operate in IR.

New Data Sources

The Oracle BI Server provides connectivity to a number of data source types that IR can not query directly, such as Oracle OLAP. Please refer to the current Oracle Business Intelligence Enterprise Edition support matrix for up-to-date details. The support matrix can be found at http://www.oracle.com/technology/documentation/bi_ee.html.

Benefits for Desktop Users of IR Studio

Because the definition and execution of all these Oracle BI Server features takes place outside of the BQY file itself, you can use the Oracle BI Server to implement the same feature set even for desktop users of the Interactive Reporting Studio. Instead of providing one or more database connections directly to the data sources, remove all database middleware and replace it with the Oracle BI Server ODBC driver. This ensures that all data access goes through the security, data modeling and computational definitions found in the Common Enterprise Information Model.

IR Studio users can still create their own OCE connections to local data (such as Excel and MS Access files) and build their own queries and reports. They can even join this data with enterprise data controlled by the CEIM. Their method of working with BQY files only changes in respect to which OCE they use to connect to enterprise data.

Simplified Desktop Data Access Deployment

With "normal" IR Studio users, a system administrator needs to install and configure database middleware software for each and every data source a user may want to access. The actual configuration may even vary from workstation to workstation due to network topology and domain membership differences. This can create a costly, time-consuming and error-prone management situation.

Since the Oracle BI Server provides a single point of access to all available network data sources, desktop configuration is greatly simplified. Even for environments with dozens or even hundreds of data sources, the workstation only needs to install and configure a single ODBC driver to connect to the Oracle BI Server. The Oracle BI Server resides in a single location within the network, so setting up those multiple data sources at the server is also simpler because it only needs a single reference point in the network to properly connect to the back-end data sources.

Row-level Security

Since all data access rules are defined at the Oracle BI Server, administrators can insure that even desktop IR Studio users have data security rules applied no matter how they run the query, or whether they apply appropriate data filters or not. This capability to "lock down" what data is seen even for power users is especially important for applications where access to "personally identifiable information" falls under regulatory compliance with standards such as HIPAA in the U.S.A. and the 2002/58/EC (E-Privacy) Directive in the European Union.

Visibility of Tables and Columns

Topic visibility is controlled at the Oracle BI Server, not at the individual BQY document. There is no way for an IR Studio user, therefore, to circumvent the subject area visibility controls that are defined in the Common Enterprise Information Model.

For example, say the Oracle BI Server has subject-area visibility set as shown in Figure 10 above. An Interactive Reporting Studio user connecting to the Oracle BI Server via an ODBC-based OCE might login as "Administrator". They would see the "table" structure shown on the following screen:

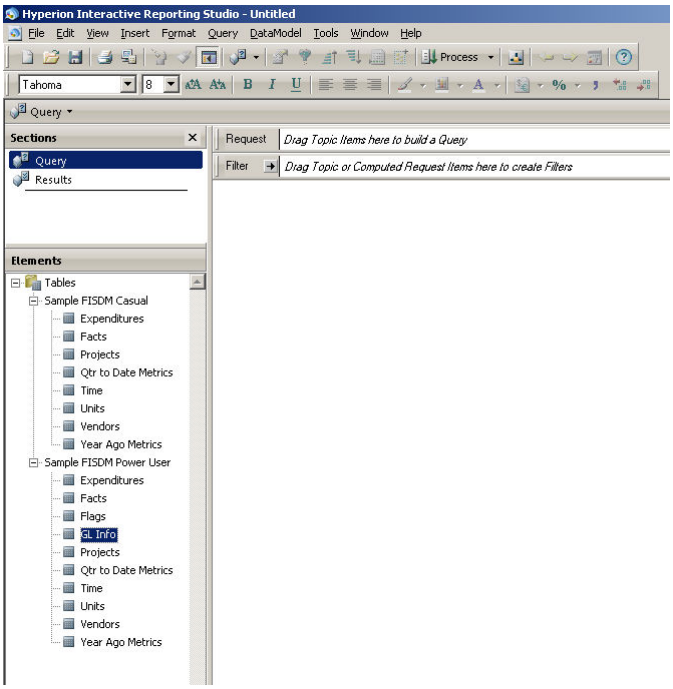


Figure 18 - Logging into an Oracle BI Server OCE as Administrator, displaying multiple Subject Areas

Another user might open the same BQY using the same OCE but log in as "User1". This user account is a member of the Casual Users group, so they will see the following subject area:

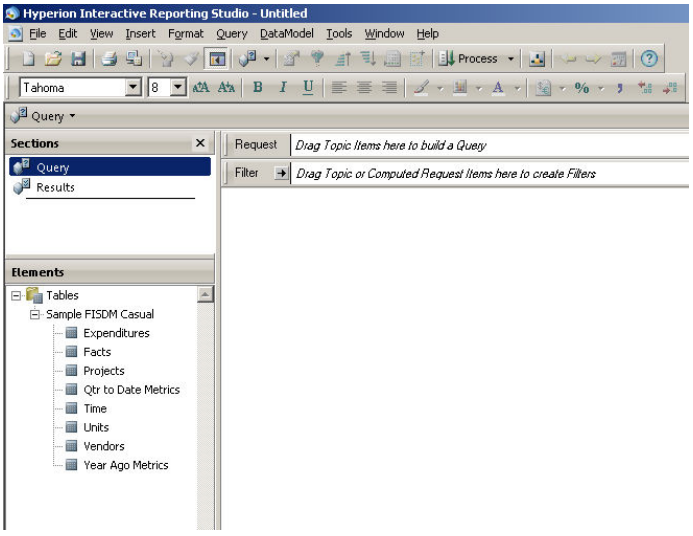


Figure 19 – Second user logging in to the same Oracle BI Server, displaying only 1 Subject Area

Implementation Steps

Interactive Reporting users who wish to leverage these new Oracle BI Server features do not need to start from scratch. A number of features and utilities exist that can ease the transition from direct connection of IR reports to their data sources to the use of indirect connections via the Oracle BI Server.

Initial configuration of the Oracle BI Server

Once you have installed the Oracle BI Server, you then need to create the skeleton of the Common Enterprise Information Model you will use for your IR reporting documents. The Common Enterprise Information Model is stored in what is known as an RPD file. This is a file created by the Oracle BI Server Administration tool. The file itself resides on the Oracle BI Server box in the \OracleBI\server\Repository file directory.

Creating the initial RPD file

The steps for automated creation of the RPD you will use with your IR reports require that the physical layer of your RPD be set up manually. Once the physical layer is defined, the first pass creation of the semantic object layer and presentation layer can be automated by the Translation Workbench utility.

Preparing to create the RPD

To create an RPD that will support all your existing and future BQY reporting documents, you first need to know what all the data sources are that will be used. You can use the EPM System 9 Services Administration Tool or EPM System 11 Configuration Management Console to view the Data Sources tab for the Data Access Services to discover what data sources are currently used by BQY documents published to the EPM Workspace. For local BQY reports, you will need to do a survey of the OCEs used and the target databases to which they connect.

Next, you need to make sure that the Oracle BI Server box will be able to reach all those data sources. You will need to make sure that all required database middleware connectivity drivers are installed, configured and tested on the Oracle BI Server box, and that any needed ODBC data source names are created as System DSNs. Make sure you resolve any connectivity problems before proceeding to the next step.

Building the Physical Layer

To start the process of creating the skeleton RPD, launch the Oracle BI Server Administration tool and select File | New to create a new RPD file. You then select File | Import | From Database for each relational and flat file data source used. Use the From Multi-dimensional option for Essbase, MSAS and other OLAP sources.

Note that you are not importing data from this data source. You are only importing the metadata about the data source. You will be presented with a dialog box that displays all available schemas, tables, etc. You can select some or all of the elements displayed to import.

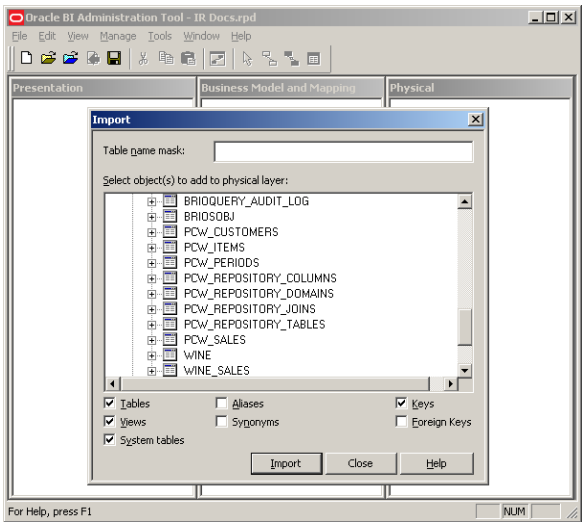


Figure 20 - Importing a data source into the RPD

Oracle recommends that you import Tables, Views, Aliases and Keys. If you need to use server-located flat file data sources such as Excel spreadsheets or Access databases, you will also need to import System tables. Do not import foreign keys. The Translation Workbench will determine the proper table-join logic based on the metadata in your BQY files instead.

For each imported data source, you will see a series of objects created in the Physical Layer.

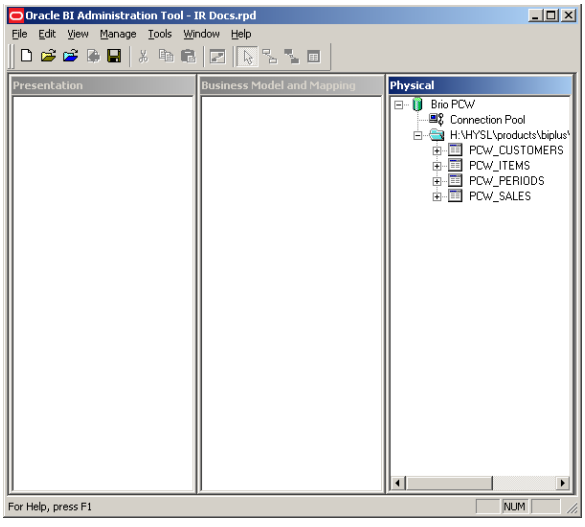


Figure 21 - Brio PCW imported into the RPD

Repeat this File | Import | From… process for each physical data source that is to be used by your BQY files. Once all data sources are imported, you are ready to proceed.

For more details on how to create the RPD's physical layer and all the available options, please refer to chapters 2 through 4 in the OBIEE Server Administration Guide, which can be found at http://www.oracle.com/technology/documentation/bi_ee.html.

Creating the full preliminary RPD

The next step is to harvest the metadata of all your BQY documents to flesh out the rest of the initial RPD design. You do this via the Translation Workbench utility.

Supported Capabilities of the Translation Workbench

The Translation Workbench is like any automated tool. It provides a rapid, wizard-based process for generating a first-pass design for your RPD but it does not support every capability you might want. Some tasks will require manual intervention after the Workbench has generated the initial RPD. The Translation Workbench does, however, greatly reduce the number of manual tasks required, especially for environments with hundreds or thousands of BQY documents to be analyzed.

As of the date of the writing of this whitepaper, the Translation Workbench support matrix is as follows:

FEATURE	SUPPORTED
RDBMS as data source	Yes
Standard SQL Queries with or without filters	Yes
Appended Queries with or without filters	Yes
Regular Filters and Sub-query filters	Yes
Table and Result sections with computed columns translated	Yes
Pivot / Chart computed columns moved & translated	Yes
IR scalar functions and simple computational expressions	Yes
IR aggregate functions	Yes
JavaScript / IR complex local computed expressions	Partly
Cubes as data source	No
Dashboard and Report sections	No
Complex Queries built with Custom SQL	No
Document Scripts	No
Reports built from models that reference local results	No

Generating the Semantic Object and Presentation Layers

The Translation Workbench is a multi-step wizard that helps automate the creation of the semantic object and presentation layers of an RPD, based on your existing BQY files. It will read the metadata in all the BQY documents published in an EPM Workspace server or located in local file directories. It will design the least number of semantic object layer business model stars into your skeleton RPD that are needed to support the total set of report objects. It will then create Presentation Layer objects to match what the IR user sees in their current query sections as tables and fields.

Multiple EPM Workspace folders and file sub-directories can be added to or deleted from the list of BQYs to harvest, so you have complete control over which BQY metadata will be included in the translation.

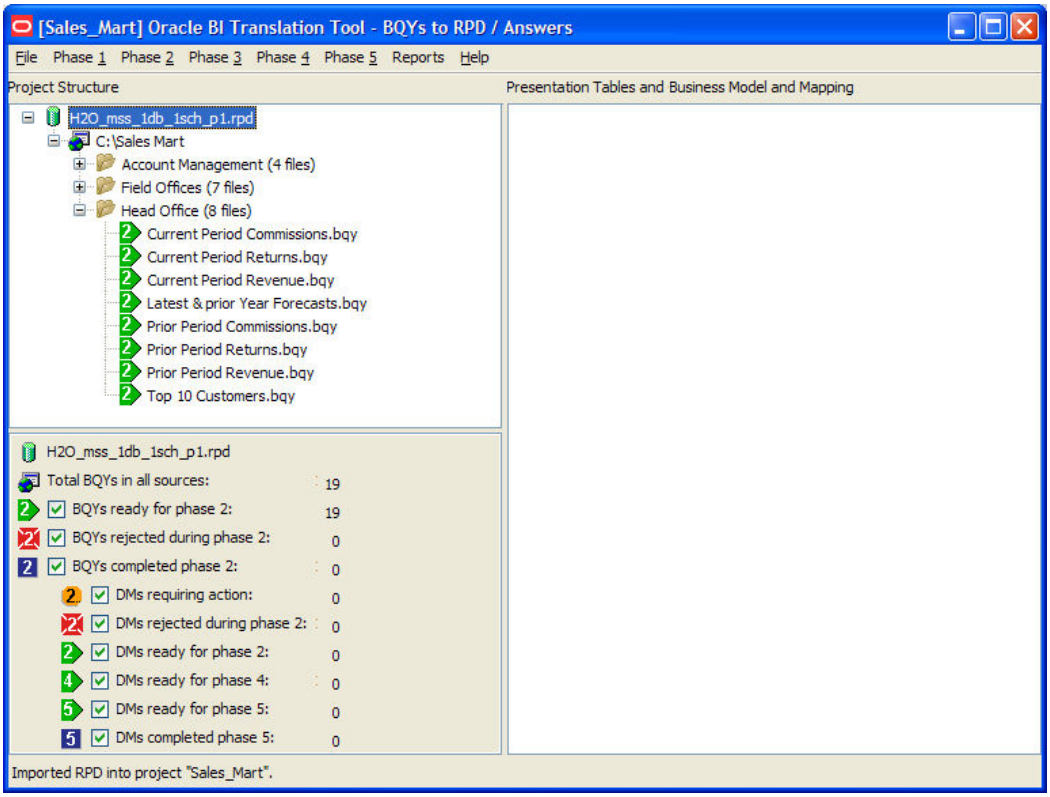


Figure 22 - Importing local BQY files into the Translation Workbench

The Translation Workbench is based on the concept of Projects. You first create a project and import your skeleton RPD. You then point to an EPM Server or file directory and import the BQY documents to be analyzed and harvested. Multiple EPM Server folders and multiple local file directories can be added to the project.

The Translation Workbench uses a 5-phase process. These 5 phases include:

1. Import the skeleton RPD created in the Oracle BI Server Administration utility.
2. Generate star schema business models based on the BQY data models.
3. Edit the standard names created (if necessary) and refine the stars.
4. Regenerate the RPD, now populated with defined Business and Presentation Layers.
    a. At this point, you manually refine the RPD design by adding hierarchies, associating level-based measures, configuring variables, etc. in the Oracle BI Server Administration utility. Refer to the details later in this whitepaper.
5. Optionally generate Answers report definitions based on BQY Chart, Pivot, Results and Table section information.

Since this whitepaper is focused on generating an Oracle BI Server RPD that will be used by Interactive Reporting, we will stop after phase 4. See the Translation Workbench documentation for further details on all 5 phases.

The wizard-style interface provides complete status information for each phase of the translation. It even provides a number of reports that document what manual user interventions will be required, both inside the Workbench wizard and post-Workbench in the Oracle BI Server Administration utility.

The Workbench is smart enough to realize that topic and field names may have been changed from the defaults and tracks the definitions back to the original physical source fields. It will default to choosing the first name encountered for items with multiple names, but will count the number of occurrences for all names used by an object. You have the option to modify the selected name by setting a Preferred Name property.

TIP: To reduce the fine-tuning of names later, Oracle recommends that your organization identify its "best practice" BQYs in terms of data model and Topic / column naming. You should then create a pivot for each model that uses these names. These pivots will be used by the Translation Workbench to determine the fact tables and to generate column names. Place copies of these "naming convention" pivot BQY files in a folder that is added to the Workbench as the first document source. This will then cause the Translation Workbench to find these names first, automatically setting these " best names" as the Preferred names.

The Translation Workbench performs a number of error and status checks along the way and provides a way for you to resolve any issues via a Problem Resolution Wizard. You will have the option to correct table roles (such as switching a table from being defined as a fact table to being a dimension), to resolve circular joins, to tag missing primary keys and to change the default visibility of columns in the Presentation layer.

You can execute the Translation Workbench in a single multi-step pass, or add new BQY files to be harvested and then re-process the appropriate steps. You can re-execute Phase 2 multiple times. The Workbench will create a new RPD each time.

Remember that the Translation Workbench is a project-based utility. By saving as a new project during various stages in your RPD development, you can snap-shot your state at various points. That way, you can look to see what kind of RPD is being generated. If you are not satisfied with a particular RPD, you can open up one of your previous "snapshot" projects and start from where you left off instead of starting from scratch.

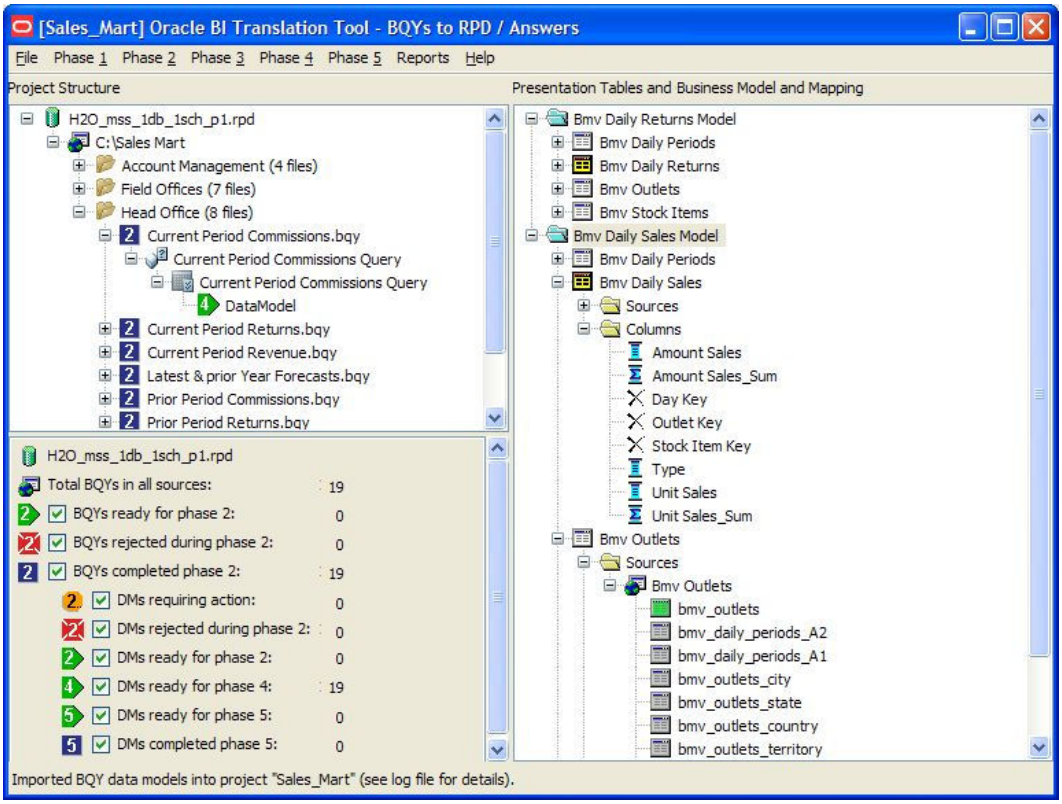


Figure 23 - The Semantic Business Models and Presentation Layer Subject Areas created by the Translation Workbench

Generating additional Aggregation objects

As the Translation Workbench harvests your BQY documents for information, it also looks at the chart and pivot sections to see what kinds of data aggregations are used in your reports. Additional logical fact columns will be created in the RPD to automatically generate the appropriate aggregated data sets as if they were actual columns in your data source. These new aggregated items can be useful in creating new BQY reports that are far simpler to design than before. The naming convention used for these new logical columns can be configured, but defaults to fieldname_aggfunc (eg, Revenue_Avg or Revenue_Max).

Your BQY documents are also harvested for what are called Level-Based Measures. For example, you may have a Results or Table section that defines computed item columns with aggregate functions for totals or sub-totals at various levels within the section, such as Sum(Revenue) or Sum(Revenue, Region). Level-based measures in the Oracle BI Server are associated with hierarchies. The creation of level-based measures in your PRD is partly accomplished by the Translation Workbench. It will create the logical fact columns that correspond to the aggregations in your Results and Tables sections automatically.

The Workbench, however, has no ability to know how to properly create the associated hierarchy since it may need to handle multiple designs. You will have to create the hierarchy and associate it to these level-based measure columns yourself after you finish with Phase 4 of the Translation Workbench.

For example, if you have a Table section with sub-totals for Revenue at the Geo, Territory and Country levels, the Translation Workbench will create Geo Revenue Sum, Territory Revenue Sum and Country Revenue Sum logical columns in the appropriate logical fact table in the RPD. Once the Workbench has created your preliminary RPD, you would then use the Oracle BI Server Administration Utility to create a MarketDim hierarchy, defining Geo, Territory and Country as levels in the hierarchy. You then edit the properties of each level-based measure in the semantic object layer's fact table to associate it with the correct hierarchy level.

Handling of Sub-Queries

In a BQY document, sub-queries are used to filter queries based on run-time conditions. The same capability is performed in the Oracle BI Server by defining Session and Repository Variables with their associated Initialization Blocks. The Translation Workbench will generate a derived logical column in the semantic object layer for each sub-query, with the SQL of the sub-query stored as the column's description. It will be up to you to create variables and initialization blocks for each and use them in the formula of the column.

Post-Workbench Manual Tasks

While the Translation Workbench can not translate any JavaScript-based computed item columns, it will create a "placeholder" logical column in the RPD design for each. This placeholder column will have a dummy value defined, with the original JavaScript code preserved as a comment. It will be up to you to design the equivalent Oracle BI Server formula for this logical column.

As mentioned before, it will be your task to create any needed hierarchies for the level-based measures created by the Workbench and to associate them with the correct hierarchy level.

You also need to create any session or repository variables and their initialization blocks that are needed to support duplication of the functionality supplied in BQY documents using sub-query filters and non-joined queries.

Finally, if any of your BQY documents make use of multi-dimensional sources such as Essbase, you will need to import these manually to your RPD's physical layer. When imported, the Oracle BI Server will also create a first pass semantic object model for any Essbase data source. You will need to refine this model and create any needed associations in the semantic layer to your relational sources. You then may need to add new logical columns, tables and/or subject areas to the presentation layer.

Feature Enrichment via the Oracle BI Server

While the Translation Workbench will create an RPD that supports your BQY documents, this only implements part of the power of the Oracle BI Server. You will want to enhance the definitions created by the Workbench to fully realize the added features of the Oracle BI Server for your Interactive Reporting users.

Data Enrichment

Because the Oracle BI Server can take complex calculations and surface them to IR users as if they were simple columns in a table, a new realm of analytics measures can be opened for IR users. The typical end user can now leverage the calculation expertise of domain experts just by choosing a field to add to a simple query design.

Centrally Managed Calculation Definitions

Critical business metrics require consistent definition to be truly useful for all groups within an organization. Many organizations find, however, that users with complex calculation needs often use BQY documents as a kind of ETL-lite tool. The data gets exported to Excel and the real calculations are then defined as formulas in a series of "spreadmarts". As a result, organizations typically lose consistency of definition for these key operational metrics.

Since even complex calculations are centrally defined in the Common Enterprise Information Model of the Oracle BI Server, consistency and ease of maintenance are automatic. Therefore, you may find it useful to move key "spreadmart" metrics definitions into the Oracle BI Server semantic object layer instead.

In addition, since the Oracle BI Server natively supports multi-source logical tables, you may discover that you can now create multi-source calculations that were difficult or impossible to create before.

Time-Series Calculations

One of the most useful and commonly used hierarchies is a time series. The Translation Workbench will not create hierarchies for you, so you should consider creating a time series hierarchy yourself. This allows you to use the Ago() and ToDate() functions add a host of time-series-oriented metrics that would have been very difficult to implement before using just the capabilities native to a BQY by itself.

Pre-defined Aggregations

While the Translation Workbench will automatically generate logical aggregation columns for the aggregations it discovers in your BQY documents, you may want to add new ones.

Performance Enhancements

Just by using the Oracle BI Server, your IR queries will take advantage of the performance benefits of intelligent shared caching and the possibility of using 64-bit database drivers. You can enhance the RPD design to provide additional performance benefits by defining aggregate navigation logic as well.

Security Enrichment

As mentioned in the topics above, the Oracle BI Server provides a host of security and session management enhancements. Investigate which ones are most important to leverage in your environment and add them to your RPD design.

Set up the Oracle BI Server for use with IR

Once you have completed the design of your final RPD, you next need to set up Interactive Reporting to use it. This needs to be done for each Data Access Service server machine, as well as any desktop PC's used for development or stand-alone usage.

Install and configure the OBI Server ODBC driver

The Oracle BI Server is accessed via an included ODBC driver. To insure proper interoperation between the Oracle BI Server and Interactive Reporting, you must use Interactive Reporting v11.1.1.0 or higher, and Oracle BI Server v10.1.3.3 or higher.

Simply run the Oracle BI Server Client Install on each computer that needs to connect to data sources through the Oracle BI Server metadata. Verify that an ODBC System DSN has been created using the Oracle BI Server Client. This DSN is typically named "AnalyticsWeb".

For desktop IR Studio users, you may also want to consider removing all other server database middleware in order to force all server data access through the Oracle BI Server. That way, you insure that all security rules and calculations have to rely on the centrally managed configuration

in the Oracle BI Server. IR Studio users will still be able to create and use local data (MS Access, Excel, flat files, etc.) by creating local connection OCE files, but the server access will go through the Oracle BI Server.

Define an Oracle BI Server OCE

Because all server-based data access is controlled centrally by the Oracle BI Server, you will only need one Interactive Reporting OCE connection file to connect to all your server-based data sources. The selection of which data sources any particular user can access is now controlled by the security settings in the RPD, not through different individual BQY and OCE files. This greatly simplifies maintenance of OCE files.

You will need to create this one "master" OCE file in a desktop instance of IR Studio. The creation of the OCE will check the connection as part of its validation, so first make sure that the Oracle BI Server is currently running.

Create a new OCE file as you normally would. When you reach the Database Connection Wizard dialog box, select the connection software to be "ODBC" and the type of database to be "Oracle BI Server".

Figure 24 - Creating an OCE that connects to the Oracle BI Server

On the next screen, log into the Oracle BI Server connection by entering the login credentials of one of the user accounts defined in the RPD, and selecting the Oracle BI Server ODBC DSN you created as the Host (the default AnalyticsWeb DSN is used in this example).

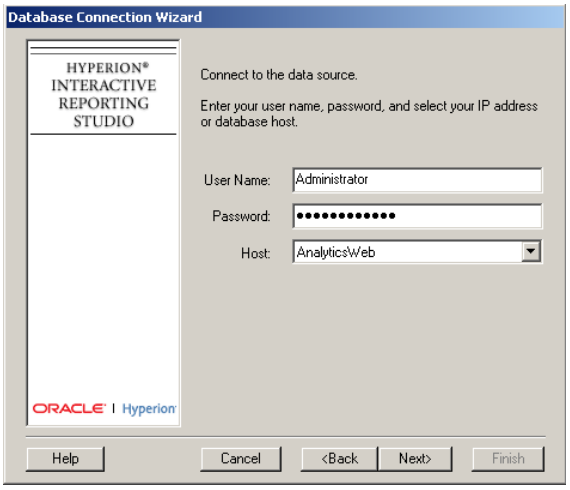


Figure 25 - Setting and verifying the connection

Set up the Data Access Server

Once the OCE file has been created, you need to import it into the EPM Workspace. Make sure that you set its access privileges such that all users of BQY report documents have at least View access. Remember that all data security should be defined in the Oracle BI Server RPD file, not in the EPM Workspace.

After installing the Oracle BI Server Client on each Data Access Service server box, make sure that the proper ODBC DSN has been created. You will use then use the Configuration Management Console to define the AnalyticsWeb DSN as a data source to each DAS for which you want access to the Oracle BI Server.

You access the Configuration Management Console (aka CMC) from within the EPM Workspace "Navigator" menu in the Administration area. You will need to make sure that the Workspace Agent UI service is running on the server first. Then, login to the EPM Workspace as a user with GlobalAdmin privileges and select Administer | Configuration Console from the Navigator menu.

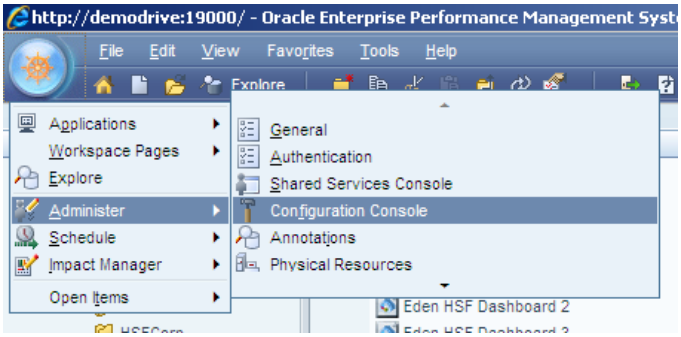


Figure 26 - Accessing the Configuration Management Console

Once you have launched the CMC, select the Data Access Service on the list in the dialog box that appears. Click the Properties icon and then the Data Sources tab to add the Oracle BI Server connection as a new data source. Configure it to use a Connection Type of "ODBC", a Database Type of "Oracle BI Server" and the data source name should be the one you created for the Oracle BI Server ODBC driver when you installed it on the DAS server machine. (The default is "AnalyticsWeb".)

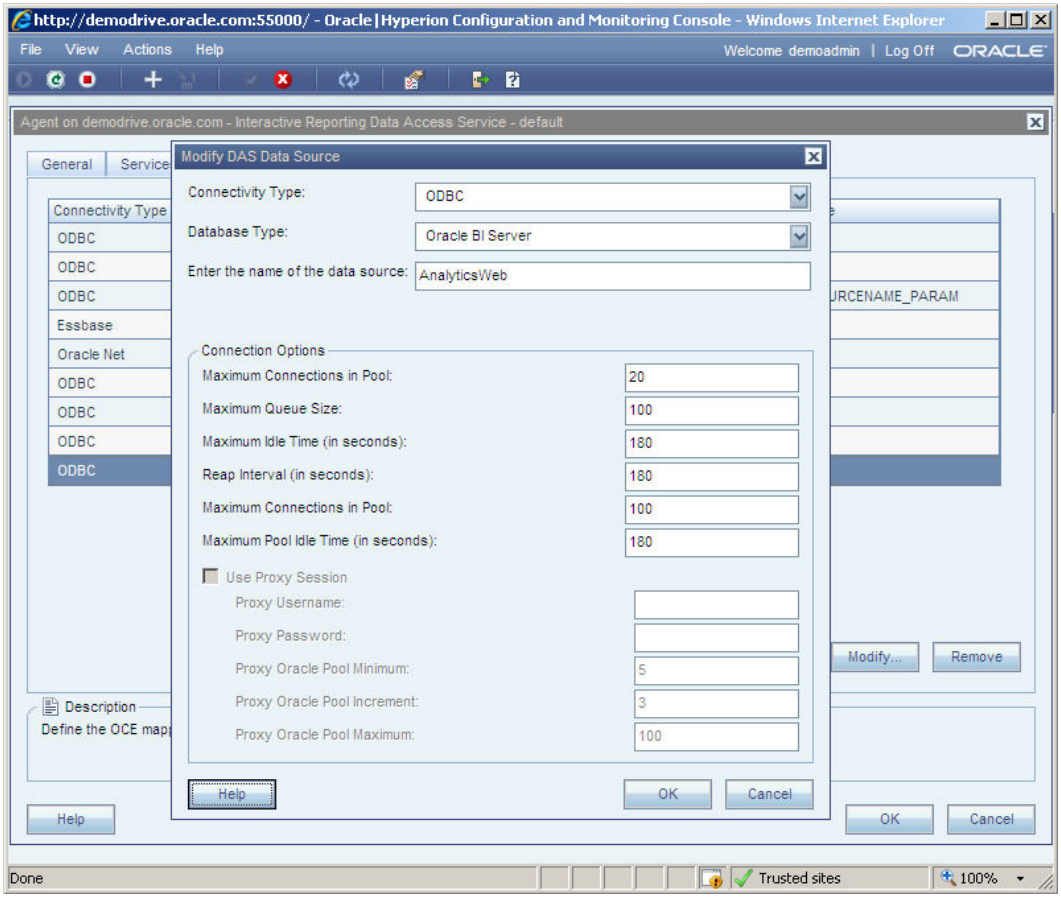


Figure 27 - Configure the DAS for Oracle BI Server access

Re-host existing IR data models on the Oracle BI Server

Using the new capabilities of the Oracle BI Server in new BQYs is easy – just use the new Oracle BI Server OCE file instead of a direct connection to your data sources. But what about existing BQY report documents that already connect directly to individual data sources?

Any existing BQY report documents can be "re-hosted" to use the Oracle BI Server instead of their current direct data connections. The utilities that exist to help automate much of this process will work on both server-based BQY files as well as BQY files stored on local file directories or network shares.

Analyze existing BQY data models using Translation Workbench

When you run the Translation Workbench through Phase 3, you will end up with a display of your BQYs grouped by common data models. Use representative BQY files from each group to analyze the data model requirements you will need to replicate in your Oracle BI Server based replacement.

Next, for each data model group, create a new BQY that uses the Oracle BI Server connection and create a new data model that contains all the columns referenced by the queries in that group. You will probably have to create meta-topics so that the names of the fields and tables in the new data model match the names of the original request line references. This new data model becomes your replacement data model.

Update existing BQY data models using Impact Manager

The Impact Manager utility can automate the replacement of old data models with the new replacement data models. It works by analyzing all BQYs in the target EPM Workspace folder or local file directory, and replacing any matching data models with the data model from a replacement template BQY.

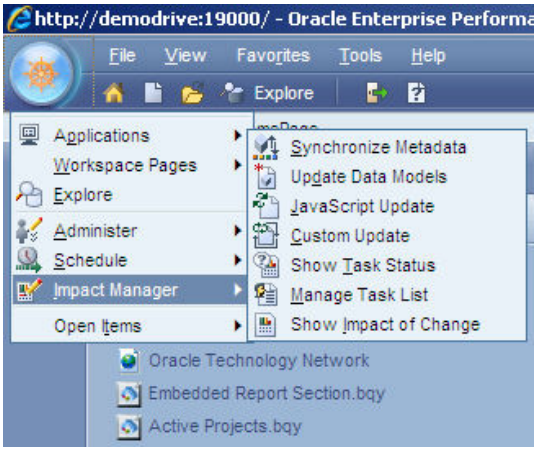


Figure 28 - Impact Manager Menu options

When run from the EPM Workspace, you can create a Task Definition File that lists which BQY files to use for repeated operations, or you can define the configuration manually in the wizard screens. Impact Manager tasks can be executed in a "run now" fashion, or be scheduled.

To use Impact Manager in the EPM Workspace to re-host your published BQY report documents, follow these steps:

1. Import all the new BQY files with replacement data models into the EPM Workspace along with the OCE that points to the Oracle BI Server.

2. Impact Manager requires that the metadata about all the BQY files has been harvested before it can proceed. Therefore, you need to re-harvest the BQY metadata so that it includes these new replacement data model BQY files. Click on the Navigate wheel and select Impact Manager | Synchronize Metadata.

3. Initiate the data model update. Click on the Navigate wheel, and select the Impact Manager | Update Data Models menu item. The Impact Manager will display the first of three configuration wizard screens.

4. In the "Specify Data Models" page, select a BQY file that represents the data model to be replaced and which data model within that BQY is the model for the replacement target. You then select the new Oracle BI Server based BQY that contains the replacement data model.

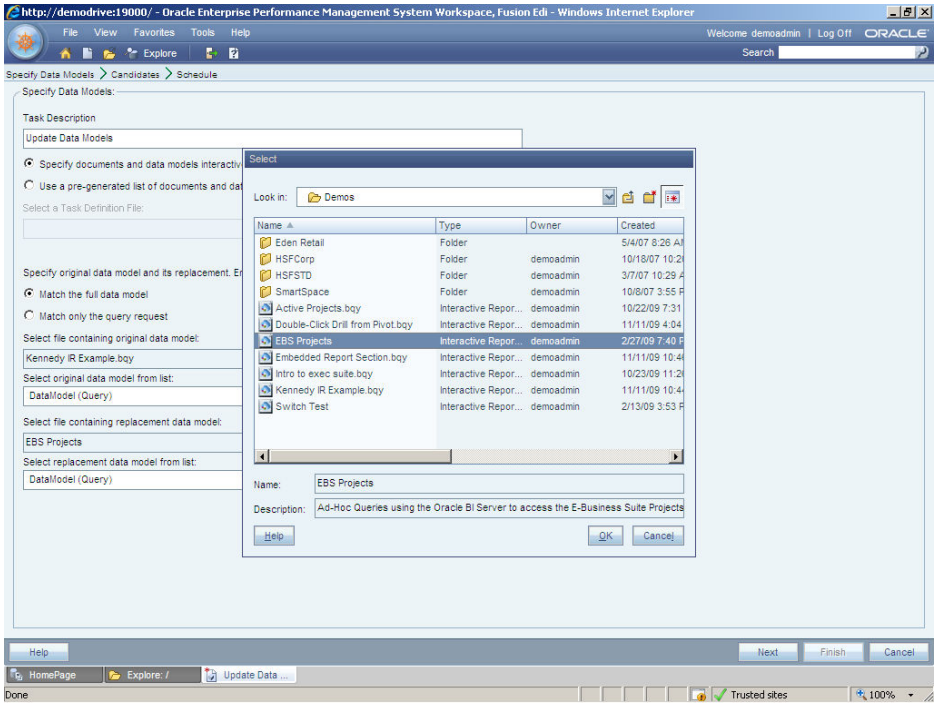


Figure 29 - Specifying the target and replacement data models

5. Impact Manager will identify the BQY files that contain data models that the replacement BQY file's data model can update. These will be listed on the "Candidates" page of the wizard. You can elect to proceed or cancel the update.

6. Finally, you will be presented with a "Schedule" page to allow you to execute the data model updates now, or schedule them for "after-hours" operation. If your list of candidate BQY documents is large, it is recommended that you choose the scheduled option.

Once the Data Model Update process begins, you can monitor the progress of all the update tasks by selecting "Show Task Status" from the Impact Manager menu list. Repeat steps 3 through 6 for each replacement data model you have created.

Additional Benefits of using the full OBIEE Plus Suite

Once you have leveraged the data models in your BQY documents to create a Common Enterprise Information Model, this can become the basis to broaden your business intelligence solution even further. Every module in the Oracle Business Intelligence foundation can leverage the Oracle BI Server and its CEIM metadata. By expanding into the complete OBIEE Plus suite, you can add yet another set of capabilities not available with Interactive Reporting, either by itself or in conjunction with the Oracle BI Server.

One Common Workspace

The additional presentation services provided by OBIEE Plus include Oracle Answers, Oracle Interactive Dashboards, Oracle Delivers, and Oracle BI Publisher. These all can be integrated into the overall EPM Workspace so end users still have one location to address all their BI needs.

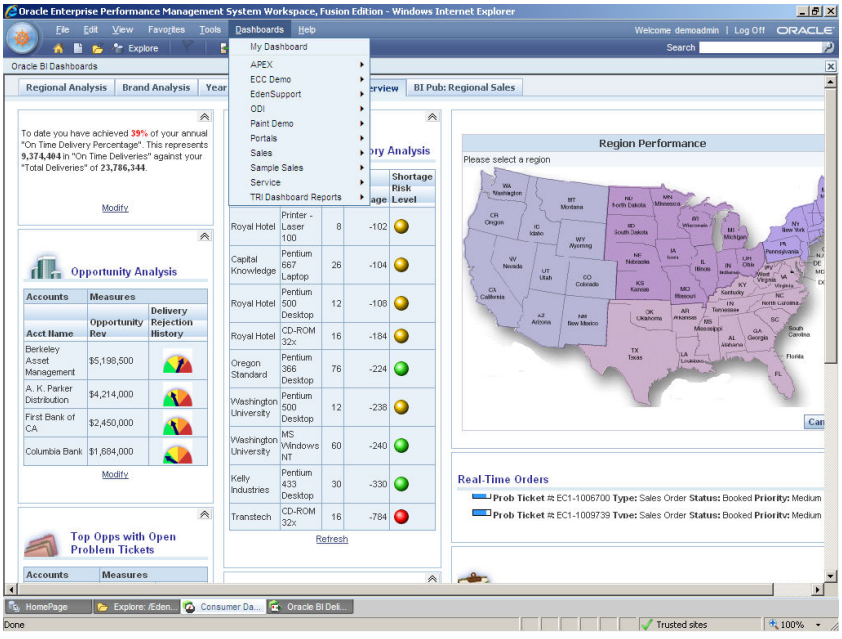


Figure 30 - Oracle Interactive Dashboards inside the EPM Workspace

Common Object Repository

In Interactive Reporting, if you wanted multiple dashboards that had some of the same content, you could try to use multiple EmbeddedBrowser objects. However, these limit the amount of cross-object control you can create, and can have a large performance impact when used extensively. So, in most cases, IR developers end up creating the same components repeatedly in multiple BQY report documents.

All the additional OBIEE Plus presentation modules exploit a repository of sharable objects known as the Web Catalog. Unlike the Hyperion document-centric approach, the objects in the Web Catalog are granular (ie, not tied to any one report document) and can be shared across multiple modules and dashboards. If a developer wants some of the same content to appear in multiple dashboards, they simply use the same shareable object multiple times. Not only does this simplify development, it also reduces maintenance overhead when designs need to change.

More than Just Ad-Hoc

While one might tend to think of Oracle Answers as merely the ad-hoc query equivalent to the IR Query section, this would be limiting it to its most basic capabilities. In addition to providing a very user-friendly, zero footprint way to create ad-hoc queries, Oracle Answers is also the basis for the creation of all the sharable objects in the Web Catalog. These capabilities include the ability to create:

- Shareable Results Sets
- Shareable Filter Objects
- Shareable Dashboard Prompts
- Shareable Presentation Variables
- Shareable Report Objects

Shareable Results Sets

When an Answers query is executed, it actually creates a uniquely identified query session object that can be addressed and (optionally) manipulated. This provides an easy way to coordinate multiple displays by simply reading, filtering or drilling into the data set in the session object. As a result, multiple Oracle and non-Oracle tools can be easily coordinated through Answers. This "results set session object" concept is also the capability that allows for integration of mapping and other 3rd party tools to be possible inside an Oracle Interactive Dashboard.

Shareable Filter Objects

Data filtering is one of the most common user tasks in business intelligence. Yet some data filters can require complex multi-layered rules. Requiring end users to remember those rules can lead to inconsistent and erroneous reporting results.

Oracle Answers allows developers to create pre-defined, named filter objects that users can simply select by name. The actual complexity of the filter rule can be hidden and user training becomes very simple – just look for the object with the filter name you want to use.

For example, a library might want to generate a set of reports about overdue books. Some will report on which books are overdue, some which report on which card holders have at least one book overdue. The definition of "overdue" varies with the type of book, though. Reference books are overdue if they have been out for more than 7 days. Regular hardcover books can be out for 2 weeks. Paperbacks and children's books can be out for a month, except in the summer, when they can be out for 6 weeks.

A developer could create a sharable filter object named "Overdue Books" that encapsulates all these rules. Users simply choose to apply this filter object, and their report will show only those items that are overdue, regardless of what kind of report they are using. And what happens if the definition of what "summer" means has to be changed? The developer goes into the definition of the "Overdue Books" filter object and updates it. Since the filter object is referencable by name, every query that uses this object will now bring back results based on the new filter rule definition.

Shareable Dashboard Prompts

Coordinated filtering of data in dashboards is also another common business intelligence task. Like the ability to create sharable query filter objects, Oracle Answers also supports the ability to create shareable dashboard prompts. Dashboard prompts can be used to apply query filters to multiple display elements, even if they are generated from different queries. The dashboard prompt will filter any display that contains a filter on the same column name in its query as the column name in the prompt. These dashboard prompt objects can be used over and over in multiple dashboards.

These dashboard prompt objects are automatically data-aware and require no coding to fill the controls with data or to apply user selected values to query filters. Dashboard prompts even include a built-in calendar widget for date-oriented fields. Simple property settings can be used to turn cascading of multiple columns on or off, and to scope the prompt to apply to only the current dashboard page, or to apply to all pages within a multi-page dashboard.

Shareable Presentation Variables

Providing coordination across multiple dashboard documents in IR can be a daunting task. Oracle Answers supports the ability to create variables that persist throughout a user's session. These are known as Presentation Variables and can be used to coordinate a host of activities.

For example, a Presentation Variable could be tied to a data entry text field and used to set a target value for user-controlled alerting.

Another common example is to use a dropdown control to set the value of a Presentation Variable, which in turn is used as a filter value in another Answers query. A dashboard can then have multiple conditional displays that are tied to whether this Answers query returns rows or not. Thus, a user's selection of a value in a dropdown control in one dashboard can determine which displays become visible on the same or another dashboard.

Shareable Report Objects

As mentioned before, all of these Web Catalog objects can be shared across multiple queries, reports and dashboards. These report objects can also have security applied to them, so that global designs can be created, and then specific security controls applied. In this manner, for example, a single dashboard design can be created that automatically "morphs" its content based on the security settings of the current user viewing the dashboard. No code writing is required.

More Robust Dashboards

The ability to leverage the sharable Web Catalog objects is merely one aspect of the robust capabilities of Oracle Dashboards.

Code-Free Dashboard Assembly

Every control in designing an Oracle Interactive Dashboards display is data-aware out-of-the-box. No coding is required to create a dashboard (even if that dashboard includes IR content). You don't even need a wizard. You use a simple drag-and-drop dashboard editor that lets you select objects from the Web catalog and place them in one or more display containers called sections.

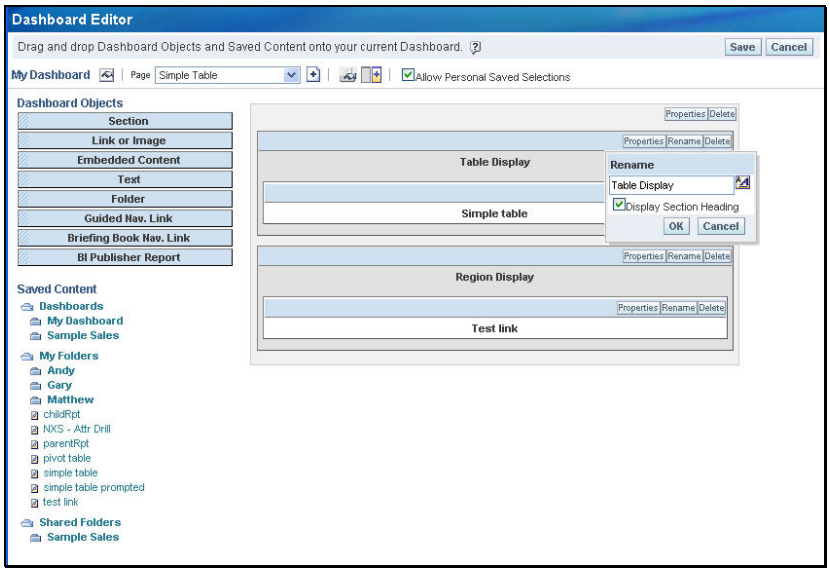


Figure 31 - Oracle Dashboard Editor

Briefing Books

Oracle Dashboards includes an oft-requested feature called Briefing Books. This allows a user to assemble content from multiple dashboards and collect them into a single "book of reports" object. A simple "Add to Briefing Book" menu item is all it takes to create one.

New Visualization Capabilities

The new presentation services in OBIEE Plus also provide a number of new ways to visualize your data. These new capabilities can enhance user understanding of their data and lead to faster, more informed decisions.

New Display Types

In addition to the standard types of displays that IR supplies, Oracle Answers supports a number of additional data presentation capabilities. These include:

- New Chart Types – Pareto, Radar and Step
- Different Bar Styles – Cylinder, Triangle, Diamond, etc.
- Chart Bands – Display one or more target or range bands behind a chart
- Funnel Chart – Typically useful in analyzing staged progress status
- Ticker – Vertically or horizontally scrolling data
- Narrative – Displays a text based description of query results, including the actual data elements as parameters (%1 for the first query column, %2 for the 2nd, etc.)
- Column Selector – Use a single report design with swappable column contents
- View Selector – Select which data display to view from a dropdown list
- Custom Legend – Create your own settings, descriptions, etc.
- Title – A separate, moveable object that contains either the name of the report object or custom title text.
- Text – A general-purpose text object that contains regular text and/or HTML markup.
- Compound Layout – A single "report" can display multiple elements on the screen, which can be arranged vertically and horizontally just by dragging them into the desired position.

Image Map Prompts

Query filters that prompt a user to select a value can be created in both Interactive Reporting and Oracle Answers. The query prompts in Answers offer a new option – the ability to define an Image Map to provide a visual way to select filter values.

For example, you might use an image of the U.S.A. and define image map areas on the graphic to determine the filter value selection to use for the Region column.

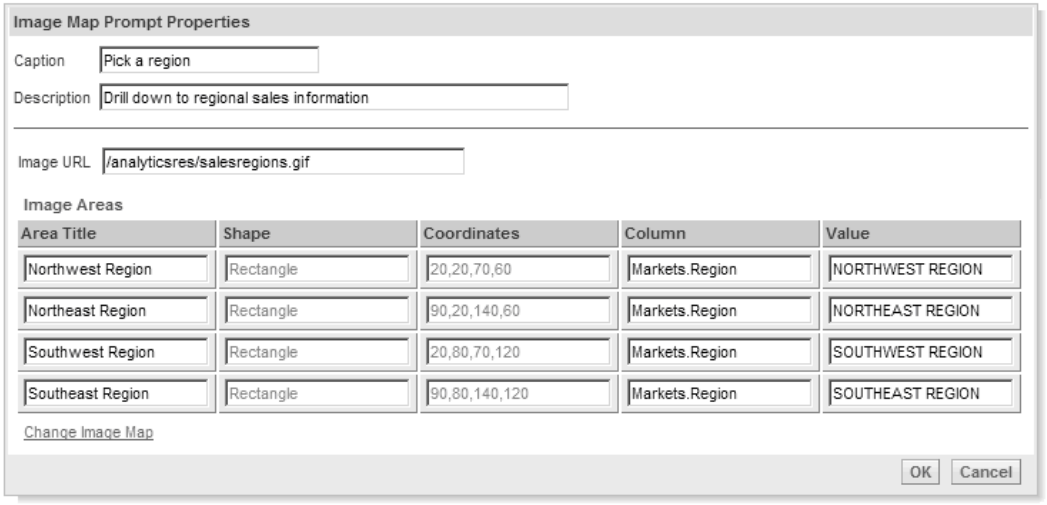


Figure 32 - Defining an Image Map Prompt in Oracle Answers

GIS Mapping Integration

The "query session object" concept allows OBIEE Plus to efficiently communicate a query data set to other presentation tools, like mapping software. Thus, you can create dashboards that combine visual, map-oriented displays alongside textual information. These displays can be coordinated, such that drilling in the map will drill in the associated data table and vice versa.

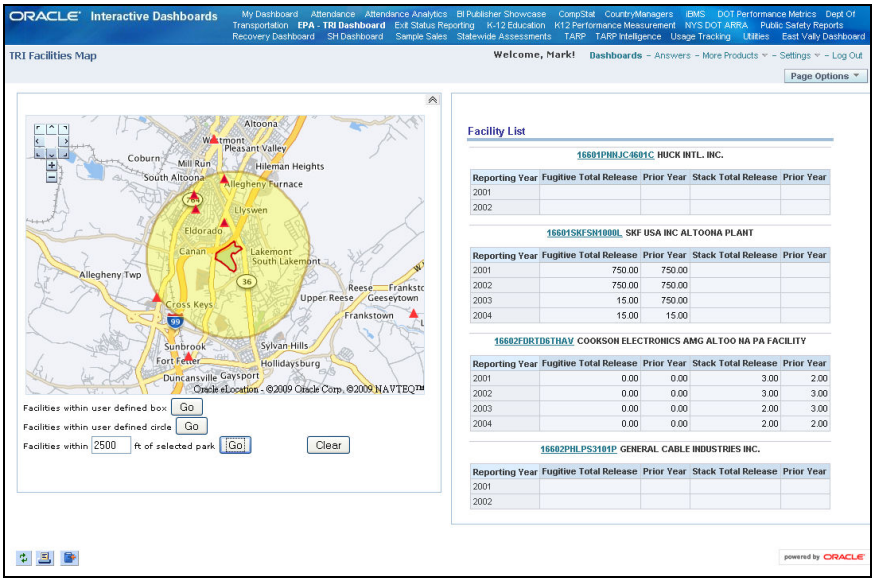


Figure 33 - Using a map-based display to select data

New Business Processes

Using the broader OBIEE Plus solution allows the implementation of more process-oriented capabilities. You can use these capabilities to make your business intelligence environment more effective and better tie it into your overall business goals and processes.

Robust Alerts

Alerting is one of the native built-in features, driven by Oracle Delivers automated agents known as iBots. These iBots support a very rich set of alerting capabilities that go beyond what the EPM system by itself can provide.

Guided Analytics

In addition to normal "something is wrong" notifications, OBIEE Plus also provides support for the next step in the alerting process – answering the question, "What do I do now?"

Guided Analytics is designed to allow a user to follow a business-oriented path to discovering root causes, next steps for action and other methods for insuring that problems get addressed in a timely and intelligent manner.

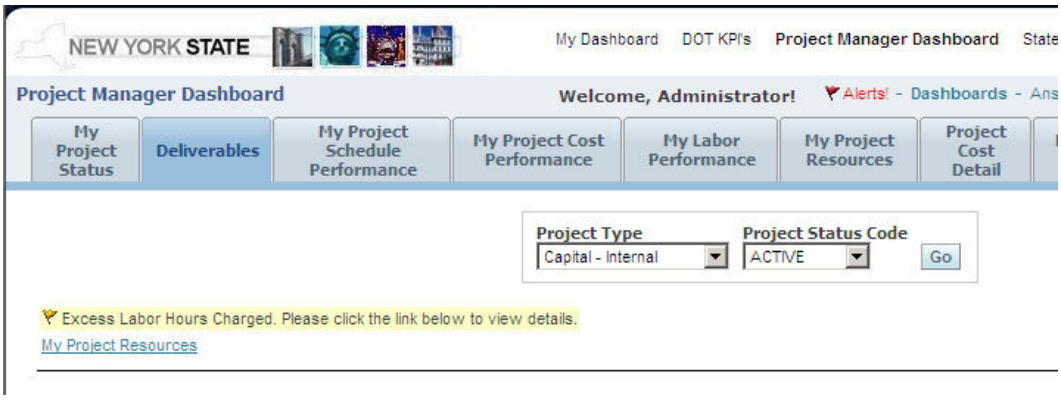


Figure 34 - Guided Analytics providing "next step" direction

Business Process Integration

That next step may also include integrating with some external business process. The OBIEE Plus platform includes integration with BPEL (Business Process Execution Language) to provide hooks into external processes.

For example, a user might be presented with a credit alert on an account that is 90 days overdue. The ability for guided analysis to tie into business processes could be used to provide a link in the report to an application that initiates a credit hold on that account. The user is directly tied into a context-based ability to act, without ever leaving their dashboard to access a separate system.

Template-Based Production Reporting

The full OBIEE Plus suite includes a module named BI Publisher which uses a template-based approach to the layout design process. This is a high-speed production reporting tool that can leverage both the queries created in Oracle Answers as well as any design templates that may already exist in your organization. BI Publisher can leverage MS Word documents, Adobe Acrobat PDF forms and other template design tools such as Flash.

BI Publisher separates the data query layer from the presentation layout. The query layer can reference one or more queries, and use one or more templates for report output. It also separates out the translation layer for supporting multiple languages, as well as the output layer itself. In this manner, BI Publisher allows the report designer to attach multiple design templates to each query, then output to multiple report formats in multiple languages with a single report object.

In many cases, users will approach the "report design team" with an MS Word document and say, "Make me a report that looks like this". Other users may forward a PDF form and ask for the form to be filled in automatically. These are all files that BI Publisher can use directly as its report template. This template approach eliminates the need to recreate existing report templates in the reporting tool itself. Simply import the template into BI Publisher and drop the database fields into the appropriate fields, tables and other objects.

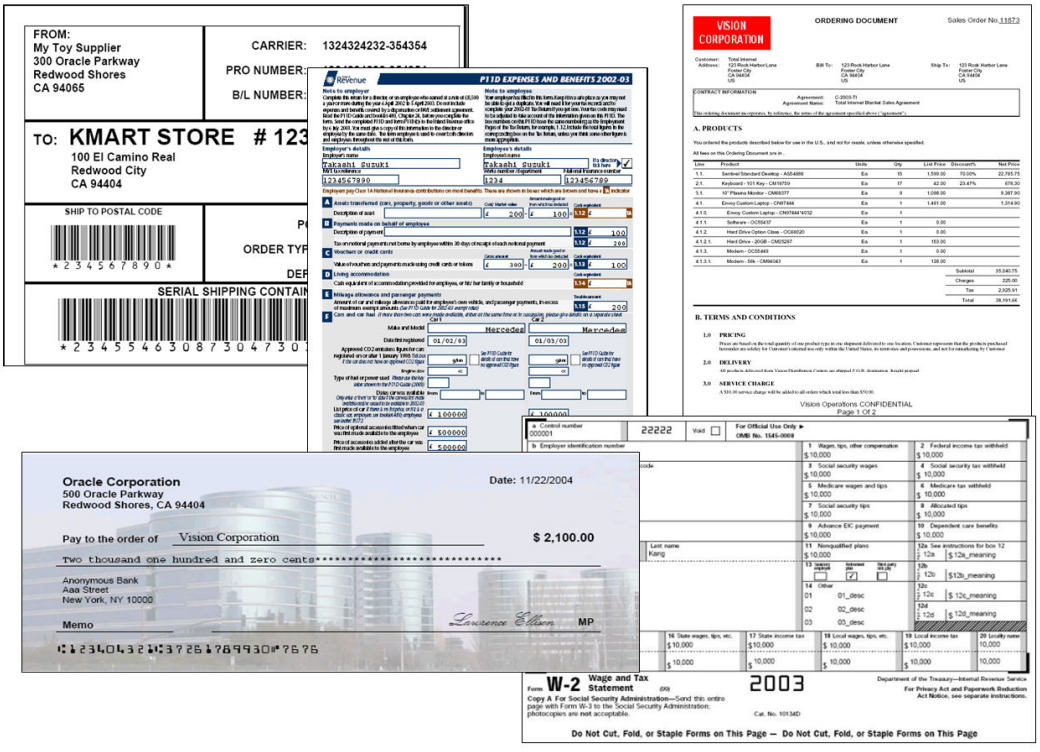


Figure 35 - BI Publisher supports a wide variety of templates and report types

Mobile Analytics Support

The broader OBIEE Plus suite provides direct support for mobile devices. Oracle Delivers provides native alerting to mobile devices, and OBIEE Plus provides the back-end support for the Oracle Business Indicators application, downloadable from iTunes.

Figure 36 - BI Applications for Mobile Devices from Oracle

Conclusion: The Best of Both Worlds

The integration of Hyperion Interactive Reporting with the Oracle BI Server provides real, tangible benefits to IR users. Existing and new Interactive Reporting BQY documents can leverage the host of Oracle BI Server capabilities transparently. So, "Brio" customers can look upon OBIEE as a way to expand the usefulness of their investment in Hyperion Interactive Reporting.

In addition, the Common Enterprise Information Model is the foundation for all the modules in the broader OBIEE Plus suite. Adding the Oracle BI Server to an IR environment can also be a first step to taking advantage of the new capabilities in Oracle Answers, Oracle Dashboards, Oracle Delivers and BI Publisher that might be useful in the future.

Using Hyperion Interactive Reporting with the Oracle BI Server
December 2009
Author: Mark Ostroff
Contributing Authors: Socs Cappas, Bryan Wise

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

0109