Faster and Smarter Data Warehouses with Oracle 11g

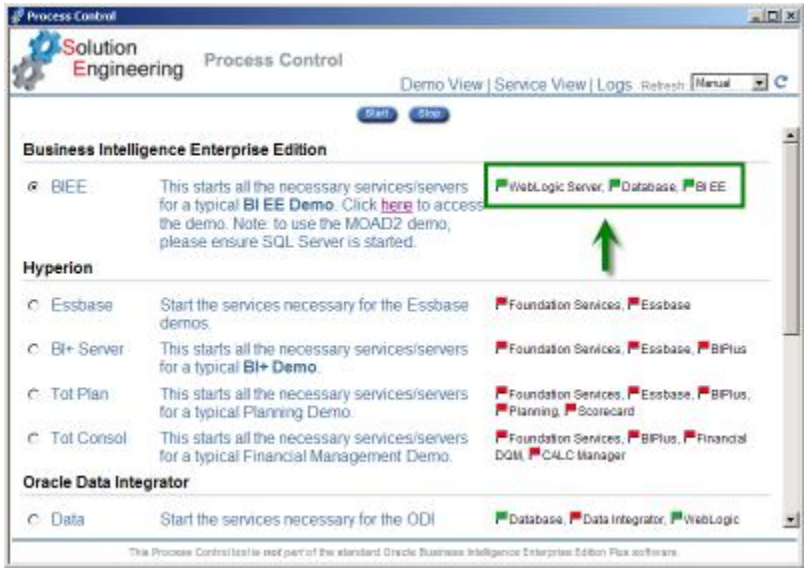
SETUP INSTRUCTIONS BEGIN HERE

Note: Please follow **all six** of the following setup instructions prior to beginning the demo. The startup of the services in Setup Instruction #4 can take several minutes. In addition, the Windows OS in the Virtual Machine may need to increase the size of its virtual memory paging file, which can take some additional time during the startup of the services.

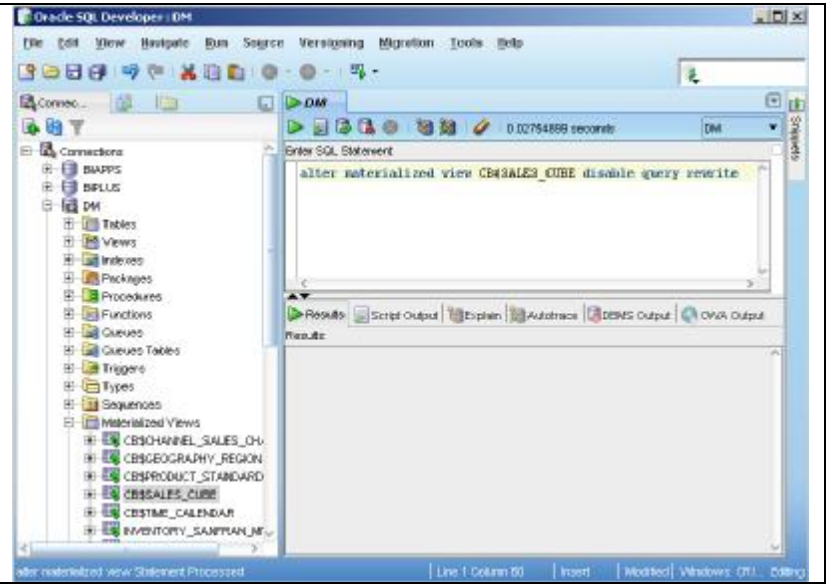
Setup Instruction #1	Start the VMWare BIC2G image, and double-click the **Process Control** icon on its desktop. Note: allocate as much memory as possible – preferably at least 1800KB – to running the VMWare image.

Setup Instruction #2	Click the **Service View** link at the top of the screen. Select **Vlamis11g.rpd** from the Repository dropdown, and **Vlamis11g** from the Web Catalog dropdown, then click the **Set** button. When the Repository and Web Catalog have been changed, you should see the selections changed as indicated with the green arrows. Note: You can either set the Refresh mode (top right of the screen) to **Continuous**, or you can leave it set to **Manual** and click the refresh icon to update the Process Control screen manually.	

Setup Instruction #3 – VERY IMPORTANT!	Open Windows Explorer in the VM, and navigate to **c:\DEMOS\Vlamis.** Double-click to execute the file named **OBIEE Vlamis.cmd**. This copies the file named Vlamis_NQSConfig.ini into NQSConfig.ini to disable caching in OBIEE. (By default, the OBIEE demos have caching enabled.)	

Setup Instruction #4	Click the **Demo View** link at the top of the screen. Select the **BIEE** radio button, then click the **Start** button. When all appropriate services have started, the top row of three flags will all be green as shown here.	

Setup Instruction #5		From the Windows Start menu, select **Programs**…**Oracle Business Intelligence** … **Presentation Services** to bring up the login screen.	

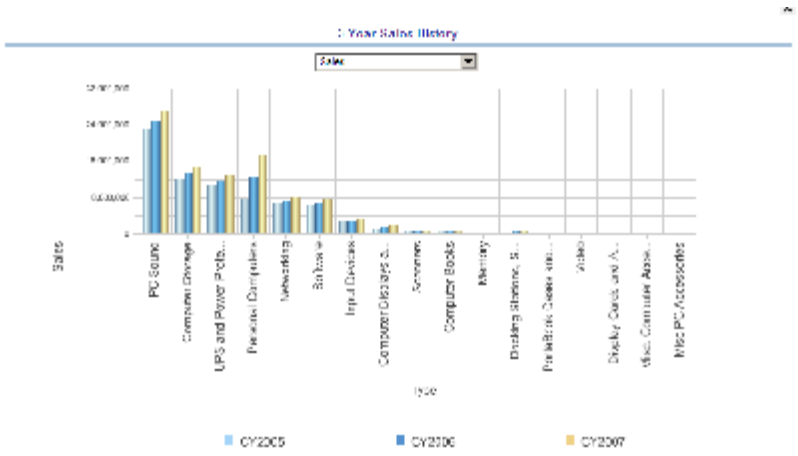
Setup Instruction #6	Start Oracle SQL Developer, (shortcut in the top section of the Start menu) and connect to the Oracle database using the DM connection. Type this command into the SQL Statement area, and press the F9 button (or mouse-click the green arrow) to execute it: **Alter materialized view CB$SALES_CUBE disable query rewrite.**	

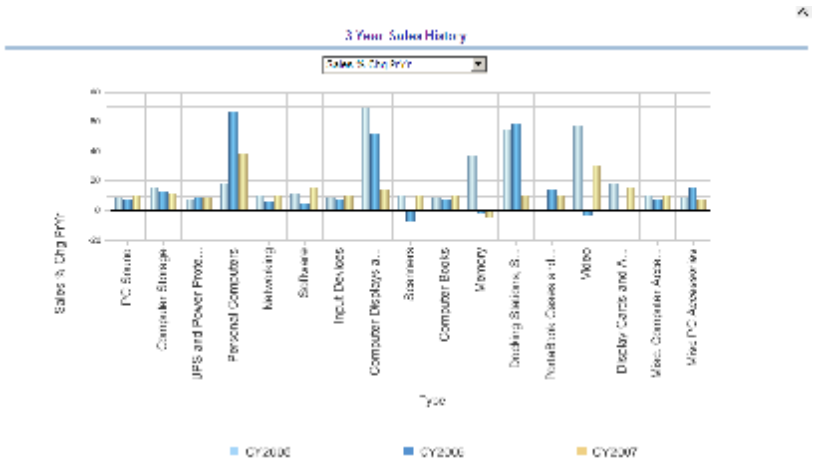
DEMO BEGINS HERE

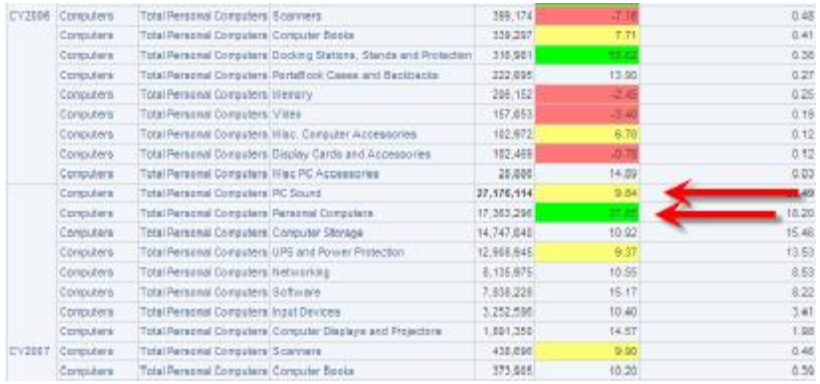
Screenshot	Script	IT Selling Message	Do This
	The purpose of this presentation is to demonstrate the use of Oracle Business Intelligence Enterprise Edition (OBIEE) in conjunction with the multi-dimensional OLAP functionality in the Oracle database. We will purposefully concentrate on the value of OBIEE and Oracle OLAP, not how to actually map the data and build the reports. This presentation uses the "DM" sample schema downloadable from Oracle Technology Network. "DM" is a fictional company that sells PCs and consumer electronics. Our first step is to log in to the OBIEE Presentation Services environment and open our dashboard.	OBIEE is a web-based delivery tool. There's nothing to install on the end users' PCs.	Enter the user id **vlamis** with password **vlamis** and click **Log in.**

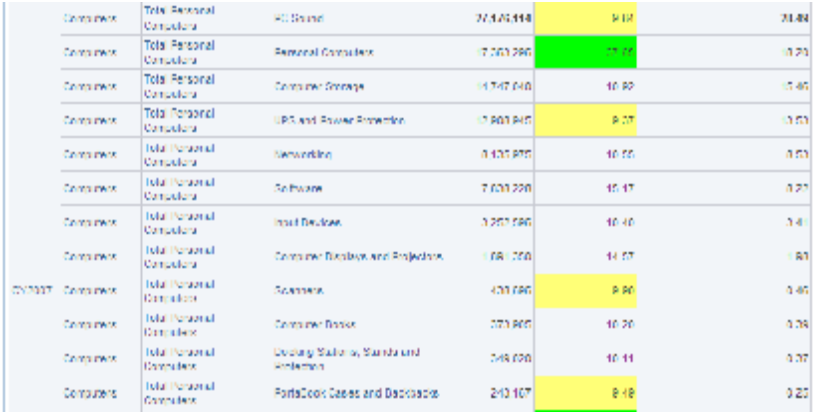
Screenshot	Script	IT Selling Message	Do This
	In the first part of this demo, we'll be generating reports such as this one, using data coming from a standard relational database. And then later in the demo, we'll allow the Oracle database to switch over and execute some of those same reports against our Oracle OLAP cubes to improve performance. We've already created all of the report definitions that we'll be showing today. In this presentation we'll be calling them up one by one in their finished form. But even though we're opening saved reports, the data being shown is live. These aren't just screenshots of something we did yesterday. We're actually running OBIEE, and we're hitting a live data source.		Nothing
	On this report, we're looking our company's performance in each of our three main product categories, Computers, Portable Music and Video, and Cameras and Camcorders, for each of the years 2005 through 2007. You can see that our computer sales have been growing very nicely, and that our other two categories are also growing, but perhaps not as much as the sales of Computers.		Nothing

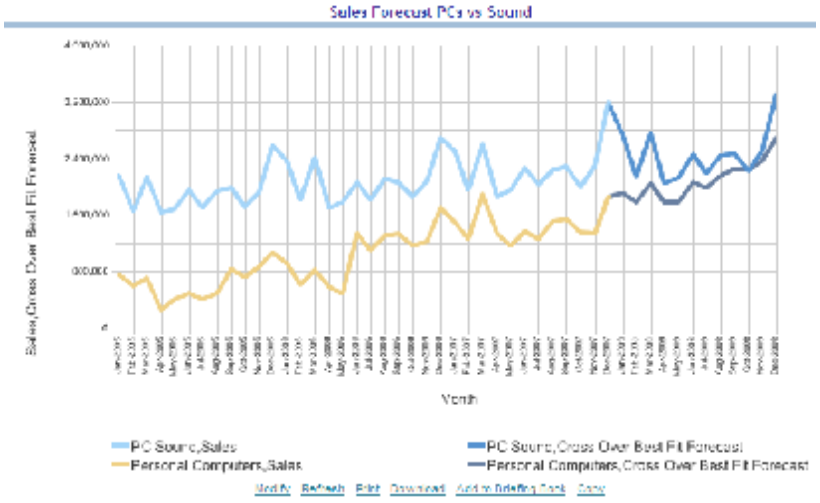
Screenshot	Script	IT Selling Message	Do This
	One of the features of OBIEE is the ability to drill into any amount on the screen, whether it's shown as a number in a table, or as a point on a line, or as a segment of a pie chart, or, in this case, a particular bar in a bar chart, If I were to click on, for example, this tall gold bar, I would be drilling into the results for Computers for 2007. You can also see that when I rest my mouse pointer on a particular bar, the actual value associated with that bar pops up.		Point to, **but do not click on**, the tall gold cylinder for Computers.
	But right now, I'm interested in seeing all of the subcategories under my Computers division for each of the past 3 years, and I can do that by just clicking the 'Computers' title just below my chart.		Click on the **Computers** label below the chart.

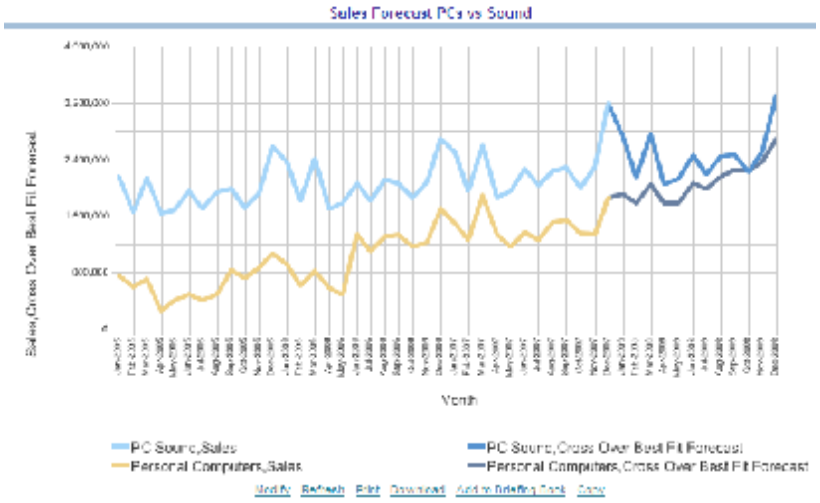
Screenshot	Script	IT Selling Message	Do This
	On this screen, I see that Total Personal Computers is the biggest component of our Computer sales, and I want to see ***its*** components, so I'll click on the 'Total Personal Computers' title under the chart.		Click on the **Total Personal Computers** label below the chart.
	Interestingly, we see that our PC Sound components, things like sounds cards, speakers, headphones and so forth, are actually our biggest sellers, and are contributing the greatest dollar amount of sales from our Personal Computers category. But is PC Sound one of our biggest growth products? Here at the top of the chart, we can choose to display different types of data. Right now, we're looking at dollar sales. But perhaps we're also interested in seeing a metric like Sales Pct Chg. Vs. Prior Year. Let's select Pct Chg. Vs. Prior Year from the dropdown and see the results.		Select **Sales % Chg PrYr** from the view selector dropdown.

Screenshot	Script	IT Selling Message	Do This
	Here we can see that sales of Personal Computers are growing at a much faster rate than are our PC Sound products, and Computer Displays are growing pretty well too. We can also switch between different types of views of the same data. If we scroll down a bit…		Scroll down to the Analysis View of the data.
Select View Analysis View Sales $ Department / Category / Type / CY2005 / CY2006 / CY2007 Computers / Total Personal Computers / PC Sound / 23,076,267 / 24,742,478 / 27,176,114 Computer Storage / 11,847,252 / 13,294,688 / 14,747,040 UPS and Power Protection / 10,870,065 / 11,802,504 / 12,908,945 Personal Computers / 7,589,574 / 12,614,328 / 17,363,296 Networking / 6,939,163 / 7,359,683 / 8,135,975 Software / 6,506,614 / 6,805,913 / 7,838,228 Input Devices / 2,740,538 / 2,946,081 / 3,252,596 Computer Displays and Projectors / 1,088,641 / 1,650,894 / 1,891,350 Scanners / 429,981 / 399,174 / 438,696 Computer Books / 315,015 / 339,297 / 373,905 Memory / 211,335 / 206,152 / 197,066 Docking Stations, Stands and Protection / 199,833 / 316,981 / 349,020 PortaBook Cases and Backbacks / 194,988 / 222,095 / 243,167 Video / 162,573 / 157,053 / 204,101 Display Cards and Accessories / 103,271 / 102,469 / 118,225 Misc. Computer Accessories / 96,433 / 102,972 / 113,371 Misc PC Accessories / 25,141 / 28,886 / 31,057 Grand Total / 72,396,685 / 83,091,648 / 95,382,155	… we see a tabular view of our data, what's known in OBIEE as a Pivot Table, with our Department, Category, and Product Type going down the page, and each of our selected years going across. On this report, we're choosing to display our sales figures. We can use one OBIEE's features known as a View Selector to switch to a different view of our data. We've set up a view that we're calling our Data Table View.	Multiple metrics and view types in the user interface. Users can create their own content, independent of IT involvement.	Select the **Data Table View** from the view selector dropdown, then scroll down so that it fills the screen.

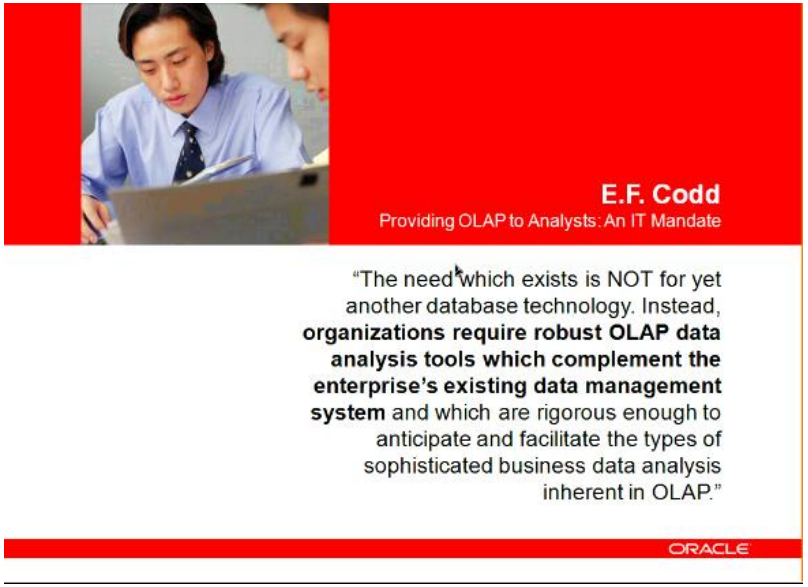
Screenshot	Script	IT Selling Message	Do This
	In this view, we can see that we've color coded our data so that for any product types where Sales Pct Chg Vs Prior Yr is greater than 25%, the sales increase percentage is shown in green, and any that are between 0 and 10% growth are shown in yellow. And if we scroll all the way down and click on this button to show us all of the records…		Scroll to the bottom, and click the All Records button. When all records are displayed, scroll down to see the 2007 results.
	…we can scroll down further and see that we've color-coded, in red, any product groups that had declining sales from one year to another. So here we see that our PC Sound sales grew by 9.84% in 2007, but our sales of Personal Computers grew by 37.65%, and so forth, and we've used color highlighting on our points of interest.		No Action

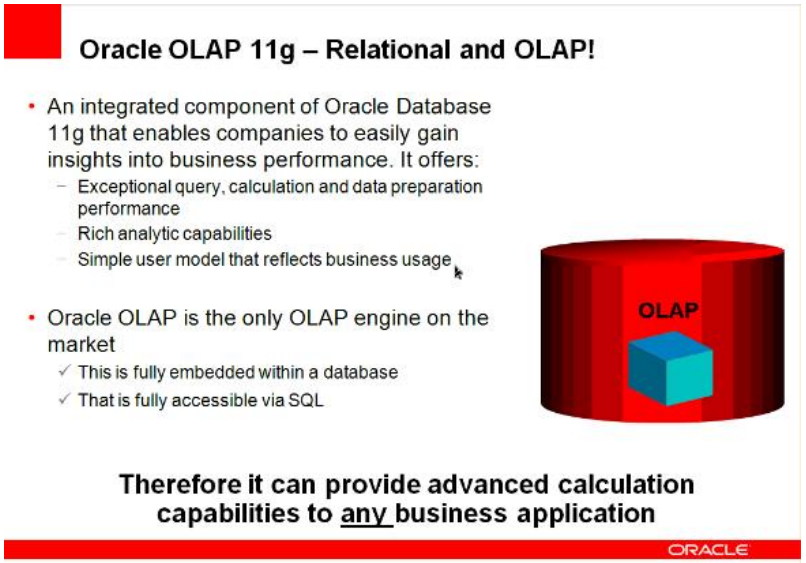
Screenshot	Script	IT Selling Message	Do This
	So far, we've seen reporting of ACTUAL results, and those can generally be produced using the features of the Oracle Relational Database. But one thing that the Relational Engine doesn't have, and one of the features of Oracle OLAP, is the ability to generate forecasts based on historical trends. So next, we're going to switch over to another dashboard page, which shows us not only our actual sales, but also our sales projections for the next 12 months.		Scroll up and click on the **Sales Forecast** dashboard tab.

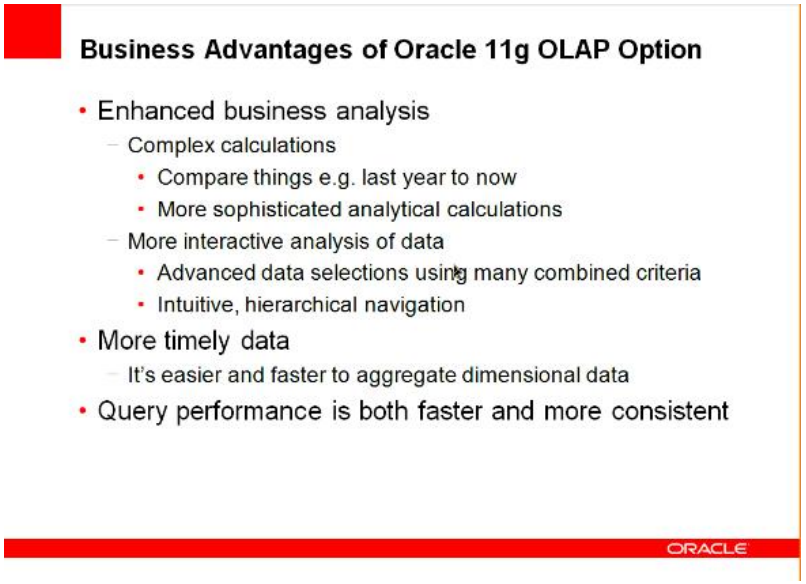
Screenshot	Script	IT Selling Message	Do This
	On this line chart, the light blue line represents sales of PC Sound products in 2005, 2006, and 2007, and the gold line represents sales of Personal Computers for that same time. The darker lines out here to the right, that are labeled 'Cross Over Best Fit Forecast', represent the forecasted sales for both of the product lines, using the "Holt-Winters" forecasting method, which takes into account not only the general trend of the data, but also its seasonality, its variability within the year. Notice that that the PC Sound line and the Personal Computers line are converging, and we should probably expect to see the sales of Personal Computers begin to surpass the sales of PC Sound products sometime in 2009.	→ → →	Point to the light blue line and the gold line, and their labels. Point to the darker line extensions on the right, and their labels. Point to the convergence of the two lines.

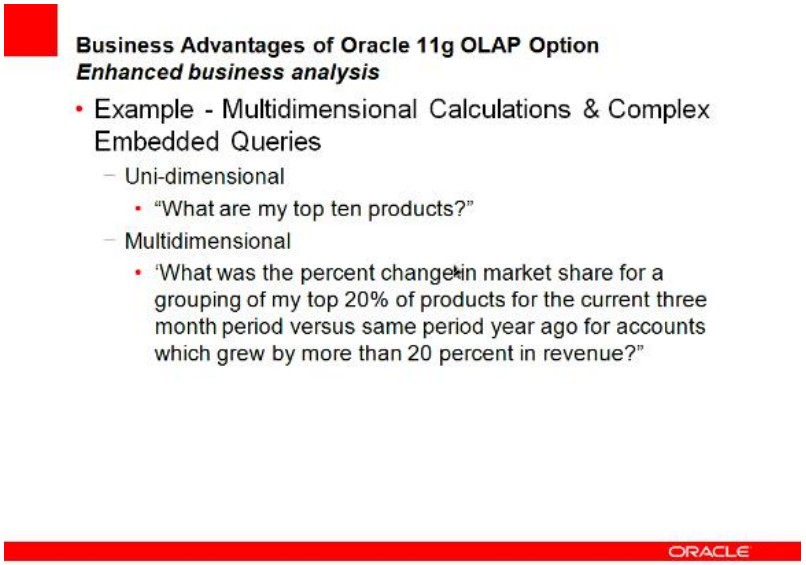
Screenshot	Script	IT Selling Message	Do This
	That's one of the powerful features of the Oracle OLAP engine, and it's built-in as part of the Oracle database. These forecasts are being generated by the OLAP engine INSIDE the Oracle database. You don't have to extract your data from the database, move it into another tool like Excel, generate your forecast, then extract it out of Excel and somehow get it back into your Oracle Database. This entire forecasting process can be set up once, completely inside of the Oracle Database using the OLAP option, and the data can be reforecasted every month, automatically.	Can combine complex functions into one environment.	Switch to the PowerPoint, and display the first E.F. Codd quote.

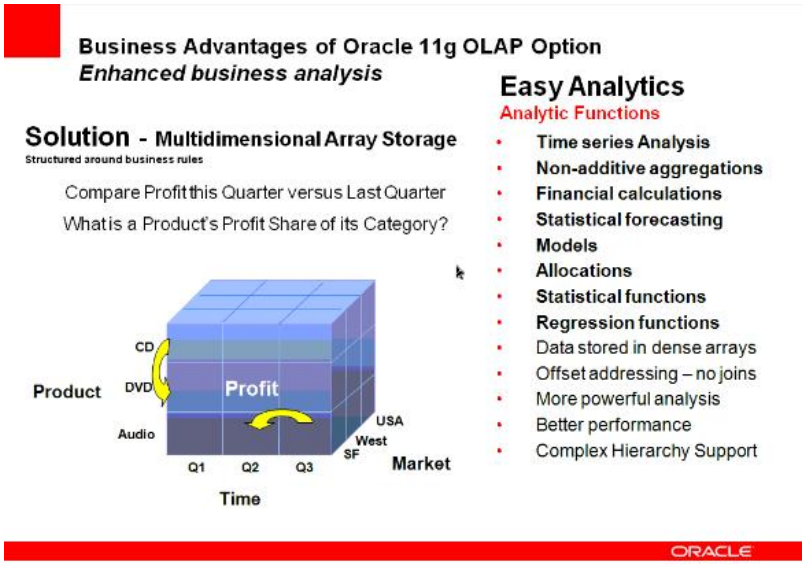
Screenshot	Script	IT Selling Message	Do This
	So, why can't features like forecasting and time series analysis just be imbedded directly into the relational engine? E.F. Codd, who is one of the founding fathers of data warehousing, gives us his take on why we haven't seen, and indeed won't see, this type of functionality from a relational engine: Codd says that "Relational DBMS products have **boundaries** with respect to providing function to support user views of data. The ability to consolidate, to view, and to analyze data **according to multiple dimensions**, in ways that make sense to one or more specific enterprise analysts at any given point in time, has been lacking." In Codd's view, purely relational databases just aren't capable of performing this type of multi-dimensional analysis because there are technological boundaries that prevent it.		Move forward to the next E.F. Codd slide.

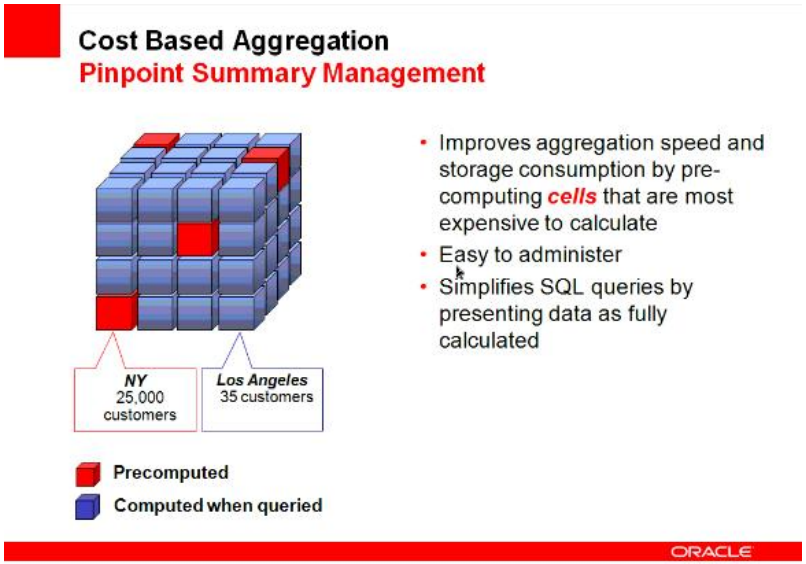
Screenshot	Script	IT Selling Message	Do This
	Codd goes on to say: "The need which exists is NOT for yet another database technology. Instead, **organizations require robust OLAP data analysis tools which *complement* the enterprise's existing data management system** and which are rigorous enough to anticipate and facilitate the types of sophisticated business data analysis inherent in OLAP." So what you need is something that can marry OLAP technology and Relational technology together, in the same environment, and you have that in the Oracle 11g database. And in fact, that functionality has existed in the Oracle database for quite some time, all the way back to 9i and then in 10g, and now in 11g, which is what we're demoing here today.	Oracle OLAP is a complementary technology that enhances the capabilities of the Relational database engine.	Next Slide

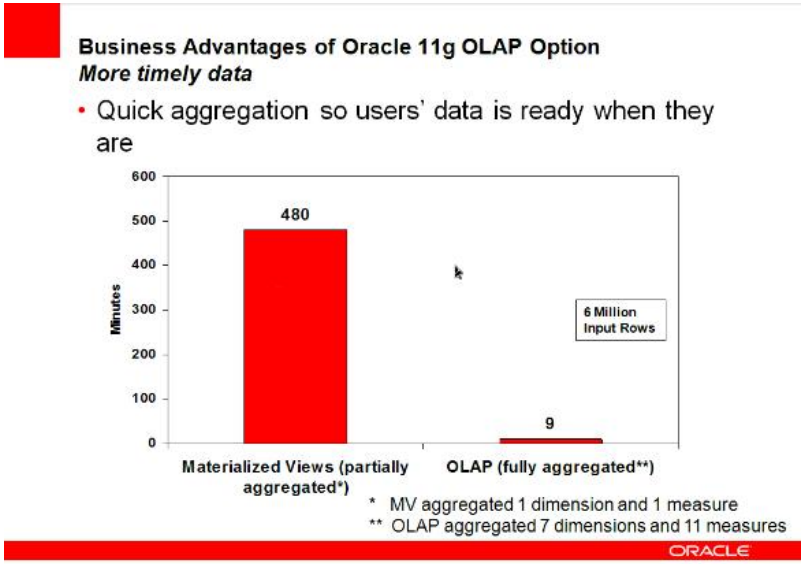
Screenshot	Script	IT Selling Message	Do This
	Oracle's OLAP technology is an integral part of the 11g database. It provides exceptional performance for queries, calculations, and data preparation. It has a comprehensive set of analytic capabilities. And it allows you to present an understandable business model to your users. Oracle OLAP is the only OLAP engine on the market that is fully imbedded within a database, and that is fully accessible via SQL. You can write ordinary SQL Select statements to access the data in the Oracle OLAP cubes, which makes them useful in ANY application which makes SQL calls to the Oracle database.	Oracle OLAP can be accessed with ordinary SQL Select statements, which opens it up for access by any SQL-compliant application.	Next Slide

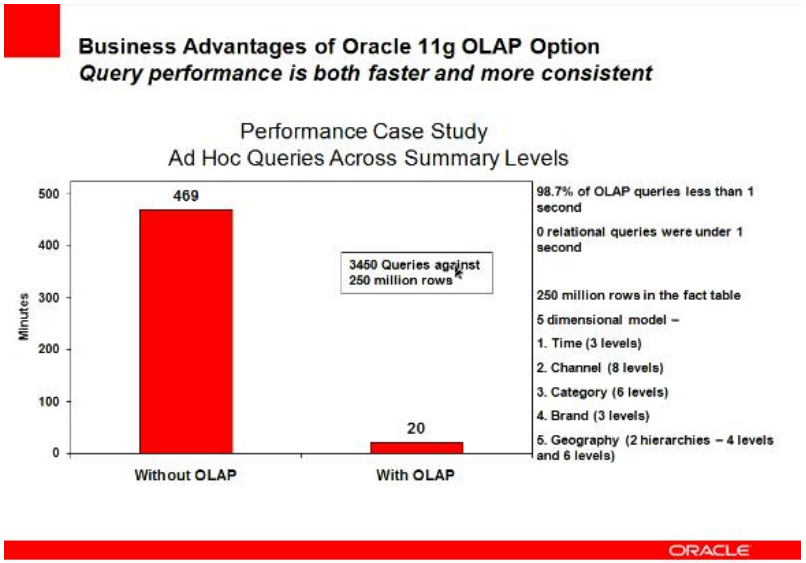
Screenshot	Script	IT Selling Message	Do This
	Oracle OLAP is particularly proficient at some of the Multi-Dimensional tasks that the relational engine struggles with. For example, the idea of time-series analysis is built-in to the OLAP engine. Comparing performance across multiple time periods is just a natural component of the OLAP environment. It has a set of sophisticated analytic functions, including a forecasting engine as we saw earlier. Users can select from multiple criteria in their queries, using combinations that might be difficult for a relational engine to parse and return results. Query performance is faster, and query times are more consistent.	No need to write very complex SQL for time series analysis when using Oracle OLAP.	Next slide

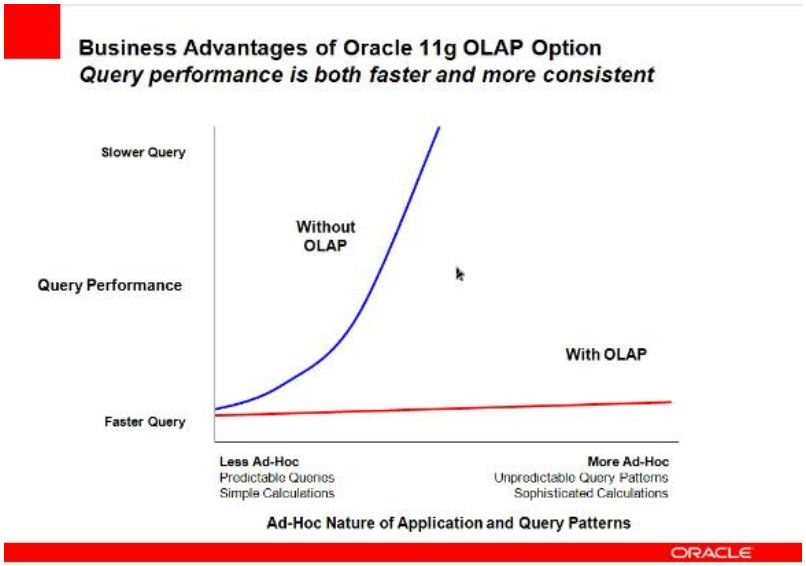
Screenshot	Script	IT Selling Message	Do This
	The OLAP option is designed for multi-dimensional analysis. Queries like this one: "What was the percent change in market share for a grouping of my top 20% of products for the current three month period vs. the same period a year ago, for accounts which grew by more than 20% in revenue?" Are really difficult in SQL. They take a long time to code, and to run, and to maintain. These types of calculations are amazingly easy for the OLAP engine to handle, and easy to set up.	Very complex queries can be handled easily by the OLAP engine.	

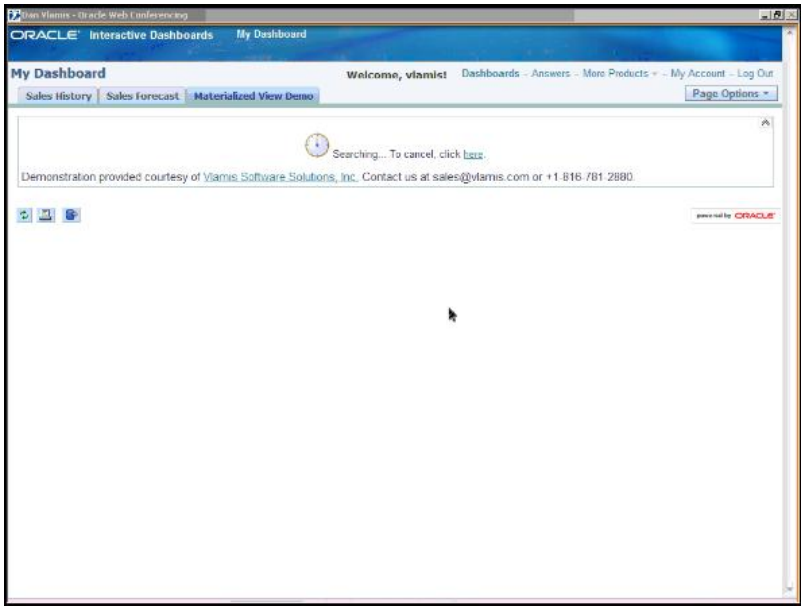
Screenshot	Script	IT Selling Message	Do This
	So, some of those key features that we touched on earlier. • Time-series analysis • Share-of-ancestor calculations • Non-additive aggregations • Financial and statistical calculations • Allocations • Regression and forecasting • Support for complex hierarchies All of these are built-in to the Oracle OLAP environment.	Very rich feature set of functions not available in a purely relational environment.	Next slide

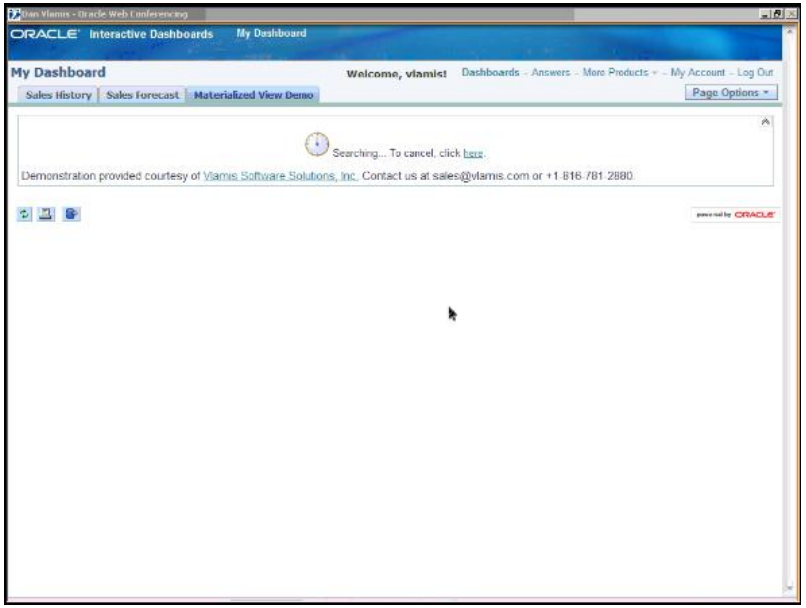
Screenshot	Script	IT Selling Message	Do This
	The OLAP option also features a Cost-based aggregation engine that allows us to specify a percentage mix of stored aggregates and aggregates that are calculated on-the-fly. For example, we might specify that we would like to precompute our 25% most expensive cells, those cells in the cube that take the longest to calculate. The OLAP engine can determine which cells those are. So if, as we see in this slide, our New York City office has 25,000 customers, and we're going to have to add up 25,000 values to see the total for New York City. But our Los Angeles office only has 35 customers, and the OLAP engine could probably calculate that one on-the-fly. So on this slide, we see that the OLAP option's cost-based aggregation engine is smart enough to precompute New York City, but to compute L.A. on the fly. But your SQL queries don't care, because the cube presents the data as fully calculated. What OLAP is doing to either retrieve data from the NY cell, or to calculate the LA cell, is totally behind the scenes as far as the SQL is concerned. And this is very easy to set up, by the way. You just specify what percentage of the cube you want to precomputed, and OLAP will figure out everything else.	The OLAP option optimizes disk storage and aggregation speed by precomputing and storing only those cells that are the most expensive to calculate at run time.	Next slide

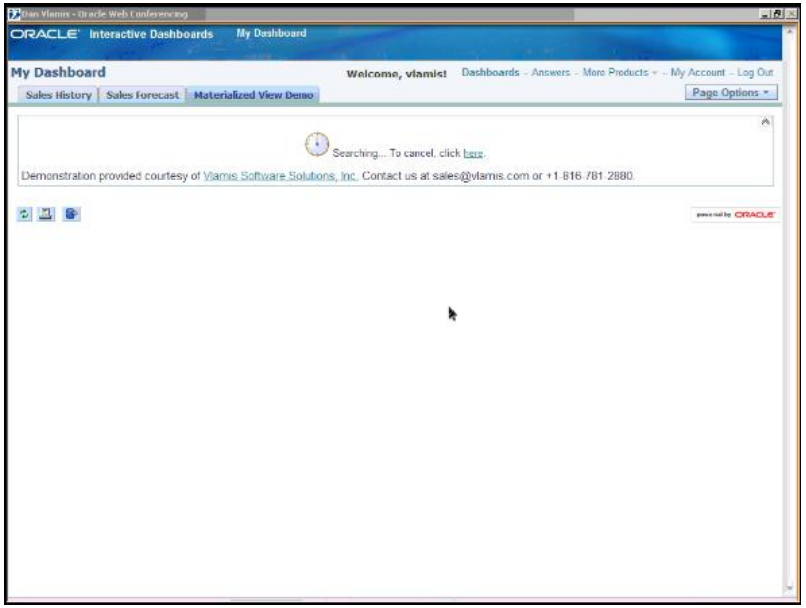
Screenshot	Script	IT Selling Message	Do This
	Oracle has done some benchmarking for the OLAP option, comparing performance of OLAP against a purely relational solution. In this chart, we see that Oracle started with a fact table containing 6 million input rows, and from that fact table they created one materialized view, that aggregated 1 measure along one dimension. And we see that it took 480 minutes – 6 hours – to complete the Materialized View. The OLAP option took 9 minutes to fully aggregate 11 measures along 7 dimensions. And you would seldom need to aggregate an entire cube. The OLAP engine is performant enough to allow you to aggregate only a portion of the cube and still get outstanding query performance.	Load times and their associated performance hits are reduced by orders of magnitude.	

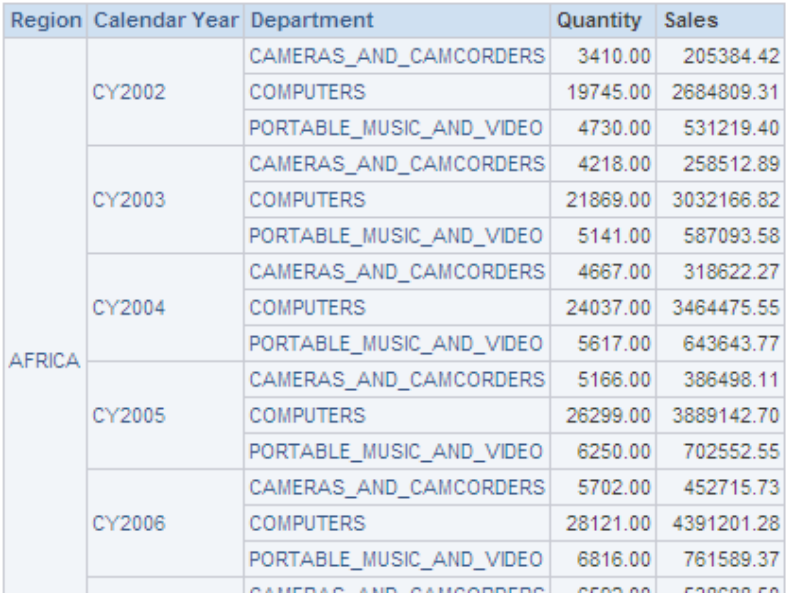
Screenshot	Script	IT Selling Message	Do This
	This Oracle benchmark measured query performance. They ran 3400 queries against a 250-million row fact table. Running just against the relational tables, those 3400 queries took about 8 hours to run. But running them against OLAP cubes, those same queries took 20 minutes. And, notice that almost 99% of those queries were returned in less than 1 second by the OLAP engine. Not a single one of the queries returned in less than a second using the relational engine.	Query performance is improved by orders of magnitude.	

Screenshot	Script	IT Selling Message	Do This
	So as you can see, query performance is both faster and more consistent when you use the OLAP option. As the queries get more and more complex, the relational engine will take longer and longer to return result. And yes, the OLAP engine is also taking longer, but it's only slightly incremental, not exponential as is the performance degradation when using the relational engine.	Enhanced query performance and consistency	Switch back to the OBIEE demo.

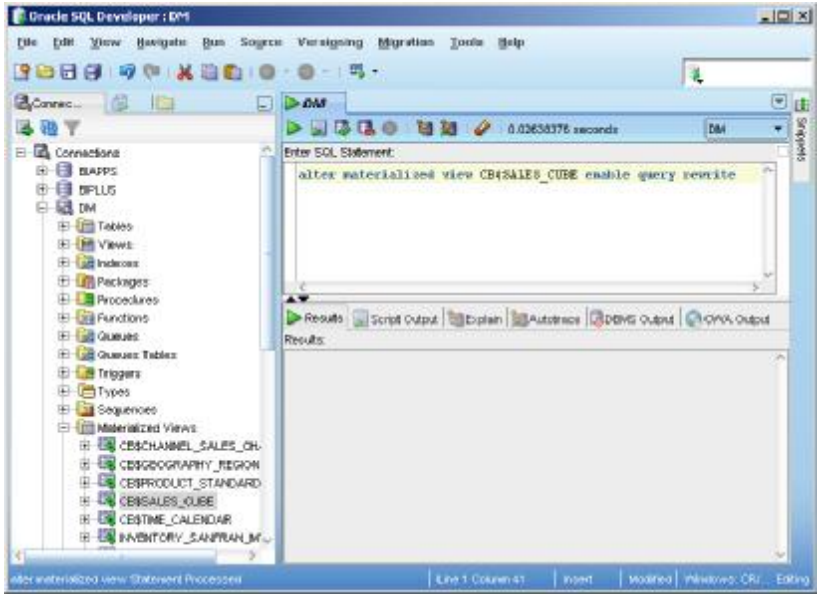
Screenshot	Script	IT Selling Message	Do This
Switching Gears again	So, how does this happen? What's the magic that gives OLAP such a performance edge? One of the key features of the Oracle OLAP option in the Oracle database, what makes it run so fast, is its ability to be used as a fully-computed Materialized View, with full visibility to Oracle's Query Rewrite engine. We've prepared another page in our dashboard, the Materialized View Demo page, and I'm going to go to that page now.		Click on the **Materialized View Demo** dashboard tab.
	This query is looking directly at our fact table, which contains data at the lowest levels of the hierarchies, so in this case that means we've stored daily data for each individual product sold to each customer through each channel of distribution on each day, and so forth. And it takes a lot of processing power and time for the relational engine to return the results. Now, we're not doing anything more than bringing back data directly from the fact table. We're not performing any difficult calculations on the data, other than just summing it up by Region, and by Calendar Year, and by Department. This query typically takes a couple of minutes, because it has to aggregate all of that data.		Keep talking…

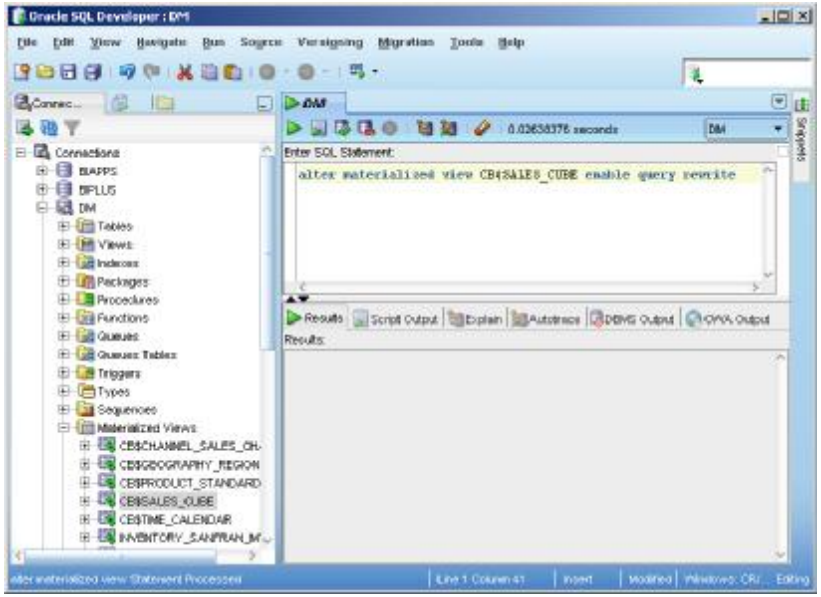
Screenshot	Script	IT Selling Message	Do This
	Now, some of you may be pretty good with SQL and you know Oracle pretty well, and you're probably thinking "well, that's dumb, why would you do that? Why not just create a Materialized View to answer this query, and you'd be right. You could just create a materialized view to answer this specific query. But there are two problems with that approach, and Oracle has solved both of those problems. The first problem is that you hade to modify your SQL to point to those materialized views. To solve that problem, Oracle has implemented a transparent **Query Rewrite** feature that automatically redirects queries to one or more Materialized Views that can provide the answers to those queries. So if you ask for a full aggregation of Sales and Quantity by Region, by Calendar Year, and by Department, the Oracle Query Rewrite engine is smart enough to know if there is a Materialized View that can be used to return results for that query, instead of always having to hit the base fact tables, and it will automatically redirect queries to use those summary tables, those materialized views, when it can.		Keep talking…

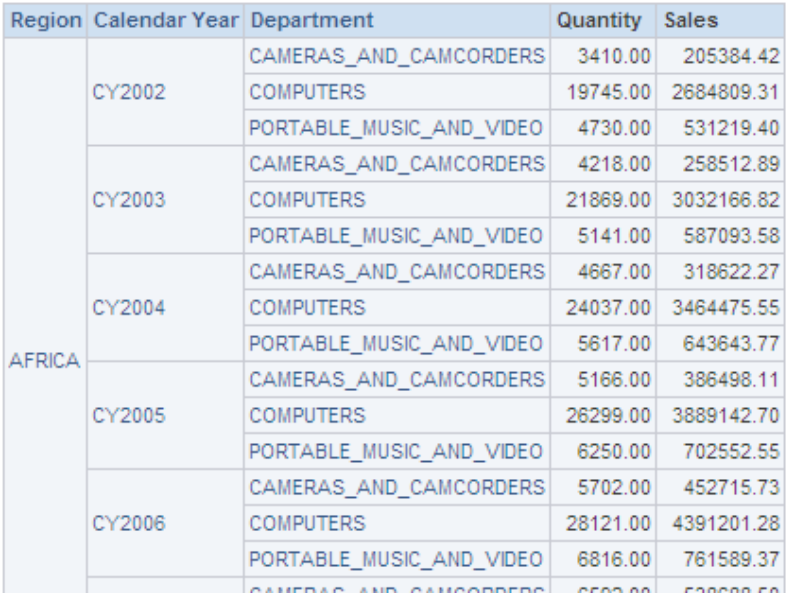
Screenshot	Script	IT Selling Message	Do This
	But the second problem is that, in order to anticipate your users' needs, you might need to create and maintain several dozen, or even hundreds or thousands of materialized views, to handle all of the combinations of all of the different levels of all of the different hierarchies of each of the dimensions. Since users aren't predictable, since they need and request information from all hierarchy levels in all dimensions in all combinations, you just can't predict where they'll go or what they'll need, so it's not feasible to try to determine which of the several hundred or thousand Materialized Views you need to create. Oracle has solved **that** problem by allowing the Oracle OLAP cubes to serve as Materialized Views, with full visibility to the Query Rewrite engine. Basically, the Oracle OLAP cubes are used as materialized views in which every cell has been precomputed. (And, as we saw earlier, we don't actually precompute every cell. We use Oracle's cost-based aggregation to precompute some cells, and compute some of them on-the-fly when they're needed.) It's like having several hundred materialized views at your fingertips, but with no maintenance, and with no need to anticipate where your users are going to go at any given time.	Creating hundreds of materialized views, and maintaining them, and refreshing them, and deciding how often are they updated, can keep one person busy full-time. All of that falls away when using Oracle OLAP as a source for query rewrite.	Keep talking…

Screenshot	Script	IT Selling Message	Do This
	As you can see, this query can take quite a while to return its results. But why should that be? If Oracle has solved the problem, why is this query taking so long? In order to give the Oracle Query Rewrite engine the ability to hit the Oracle OLAP cube, we have to enable it, and we do that by executing one very simple command in the Oracle database.		Switch to SQL Developer.

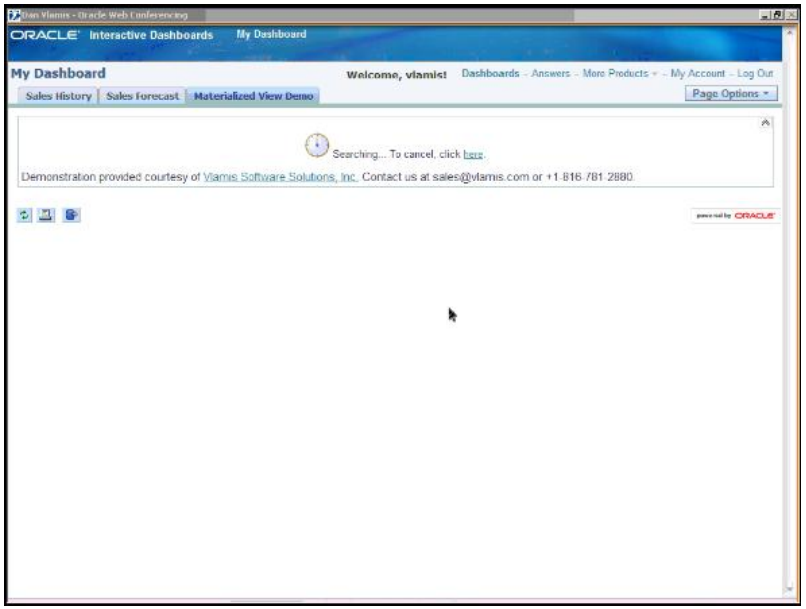
Region	Calendar Year	Department	Quantity	Sales
AFRICA	CY2002	CAMERAS_AND_CAMCORDERS	3410.00	205384.42
		COMPUTERS	19745.00	2684809.31
		PORTABLE_MUSIC_AND_VIDEO	4730.00	531219.40
	CY2003	CAMERAS_AND_CAMCORDERS	4218.00	258512.89
		COMPUTERS	21869.00	3032166.82
		PORTABLE_MUSIC_AND_VIDEO	5141.00	587093.58
	CY2004	CAMERAS_AND_CAMCORDERS	4667.00	318622.27
		COMPUTERS	24037.00	3464475.55
		PORTABLE_MUSIC_AND_VIDEO	5617.00	643643.77
	CY2005	CAMERAS_AND_CAMCORDERS	5166.00	386498.11
		COMPUTERS	26299.00	3889142.70
		PORTABLE_MUSIC_AND_VIDEO	6250.00	702552.55
	CY2006	CAMERAS_AND_CAMCORDERS	5702.00	452715.73
		COMPUTERS	28121.00	4391201.28
		PORTABLE_MUSIC_AND_VIDEO	6816.00	761589.37

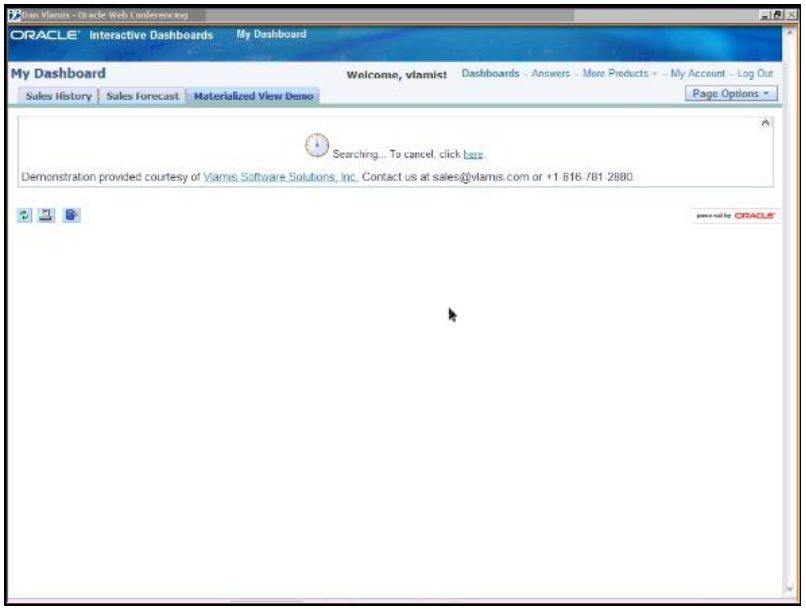
Screenshot	Script	IT Selling Message	Do This
	In SQL Developer, or Toad, we'll enter this command: **Alter materialized view CB$SALES_CUBE enable query rewrite** That's it. That's all you do to tell Oracle to use that cube as a materialized view.	Enabling queries against the OLAP option requires a simple, one-line SQL command to alter a materialized view.	Type (or revise) **Alter materialized view CB$SALES_CUBE enable query rewrite** and press F5 or click the green arrow.

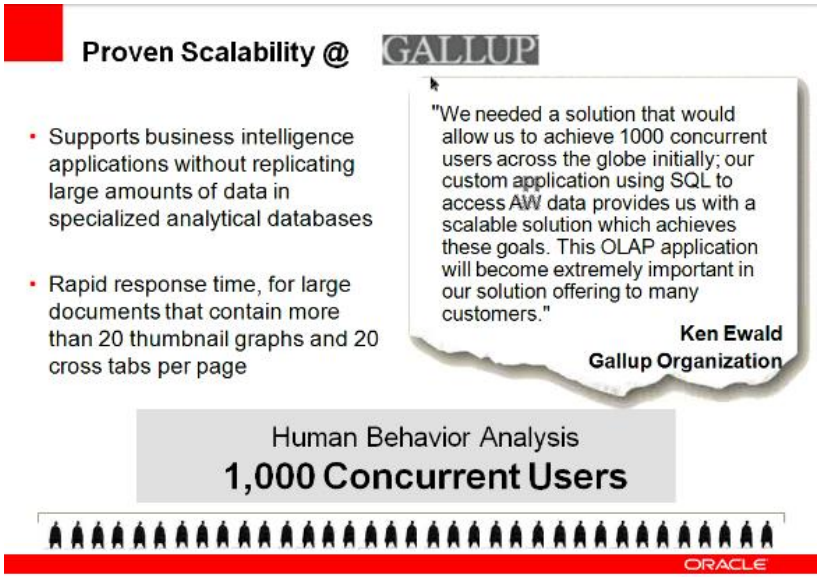
Screenshot	Script	IT Selling Message	Do This
	So now that we've enabled query rewrite against that Oracle OLAP cube, let's go back into OBIEE and reexecute our query…		Toggle back to OBIEE, and click the **Materialized View Demo** tab again.

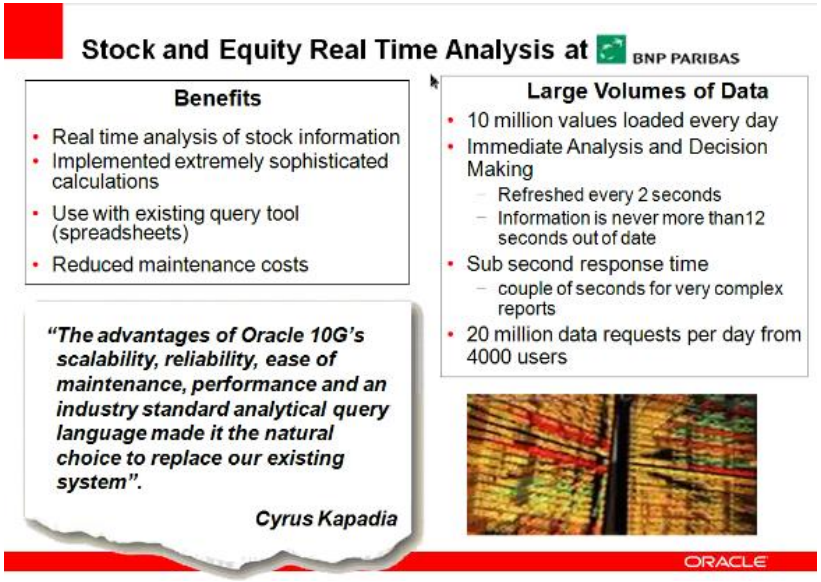
Screenshot	Script	IT Selling Message	Do This
	… and we see that the query returned in just the smallest fraction of the time that it took without Query Rewrite enabled for the cube. And just to show you that there's no smoke and mirrors going on, and no caching, I'll switch back and disable query rewrite on that cube, and reexecute it again. By the way, for this demo, I've explicitly disabled OBIEE's caching mechanism, so that every result we see is coming directly from a database query, and not from any cached results.		Return to SQL Developer, DISable query rewrite, return to OBIEE and click the Materialized View Demo tab. Tap dance (optional)…

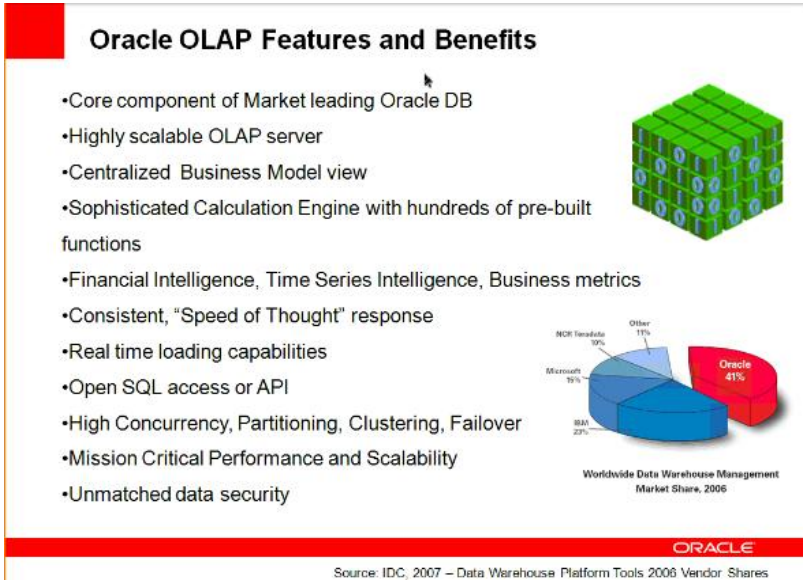
Region	Calendar Year	Department	Quantity	Sales
AFRICA	CY2002	CAMERAS_AND_CAMCORDERS	3410.00	205384.42
		COMPUTERS	19745.00	2684809.31
		PORTABLE_MUSIC_AND_VIDEO	4730.00	531219.40
	CY2003	CAMERAS_AND_CAMCORDERS	4218.00	258512.89
		COMPUTERS	21869.00	3032166.82
		PORTABLE_MUSIC_AND_VIDEO	5141.00	587093.58
	CY2004	CAMERAS_AND_CAMCORDERS	4667.00	318622.27
		COMPUTERS	24037.00	3464475.55
		PORTABLE_MUSIC_AND_VIDEO	5617.00	643643.77
	CY2005	CAMERAS_AND_CAMCORDERS	5166.00	386498.11
		COMPUTERS	26299.00	3889142.70
		PORTABLE_MUSIC_AND_VIDEO	6250.00	702552.55
	CY2006	CAMERAS_AND_CAMCORDERS	5702.00	452715.73
		COMPUTERS	28121.00	4391201.28
		PORTABLE_MUSIC_AND_VIDEO	6816.00	761589.37

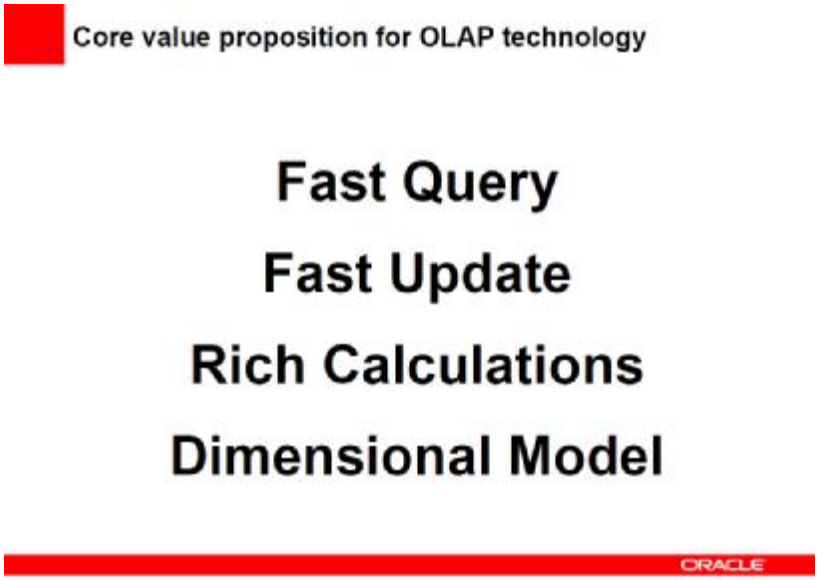
Screenshot	Script	IT Selling Message	Do This
	And you can see that, once again, it's going to take quite a while for this query to return results to us, because we disabled Oracle's ability to use our OLAP cube as a materialized view. I could switch back and forth all day and this is exactly what you would see. Slow performance against the relational table, very fast performance against the OLAP cube.		

Screenshot	Script	IT Selling Message	Do This
	Notice that I did not change the query one bit between the two different runs. The SQL that OBIEE generated isn't any different. That exact same SQL was sent to the Oracle database, Oracle passed it over to the Query Rewrite engine, and the Query Rewrite engine redirected the query to our Oracle OLAP cubes. We didn't change any SQL or do anything differently in OBIEE. It's all imbedded in the functionality within the Oracle database.		Continue optional: tap dance and wait for the query to return, then **enable** query rewrite and run the query once again. Switch back to PowerPoint slides.

Screenshot	Script	IT Selling Message	Do This
	So, how have some of Oracle's clients benefitted from using the OLAP option? Ken Ewald of Gallup states that Gallup "needed a solution that would allow us to serve **1000 concurrent users** across the globe; our custom application using SQL to access Oracle OLAP Analytic Workspace data provides us with a scalable solution which achieves our goals." So the OLAP option can help you run a system with a large number of concurrent users without causing undue strain on your hardware environment.	Excellent performance for large user populations.	Next slide

Screenshot	Script	IT Selling Message	Do This
	The BNP Paribas company needed to load 10 million rows of data every day, and they needed to load in real-time. The data was refreshed every 2 seconds, and they needed sub-second response time for 20 million data requests every day from 4000 users, to perform real-time analysis of stock information. They said that "the advantages of Oracle 10g OLAP's scalability, reliability, ease of maintenance, performance, and an industry standard analytical query language made it the natural choice to replace our existing system." So queries or environments that a purely relational engine doesn't easily support can be supported quite nicely using Oracle's OLAP option.		Next slide

Screenshot	Script	IT Selling Message	Do This
	So what's the takeaway from all of this? In this demo you've learned that Oracle OLAP: • Is a core component of the Oracle database • It's highly scalable • Provides the user community with a Centralized Business Model view of the company's data • Has a sophisticated calculation engine with hundreds of pre-built functions • Features time-series analysis, statistical analysis • Provides faster and more consistent response times • Can be accessed using ordinary SQL calls • Fully secured within the Oracle database	Oracle OLAP is fully integrated and accessible within the Oracle database	Next slide

Screenshot	Script	IT Selling Message	Do This
	In summary, we've seen that Oracle OLAP offers you Fast Queries Fast Updates Rich Calculations Dimensional Model		End of Presentation