Introduction to Functions and Variables

Functions are a way to add additional elements into your OBI Report. They enable you to manipulate data, perform computations and comparisons, and get system information. They can be simple, such as summing a range of values, or they can be very complex, using parameters and variables to return a value. Functions are a way to take a piece of data in your analysis and change it in different ways, producing new values not available simply by selecting fields.

The value of a function can be determined by input parameters, as with a function that averages a list of database values. But many functions do not use any type of input parameter, such as the function that returns the current system time, *CURRENT_TIME*.

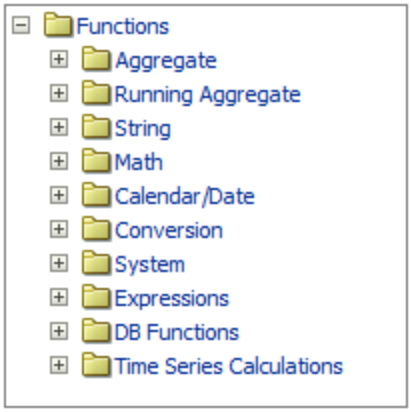
Variables are elements in OBI that store a value, such as the current fiscal year or the date six months ago. Variables can be used in an OBI analysis for filtering, for defaulting prompts, and also for use in functions.

FUNCTIONS

Functions are predefined formulas that perform calculations by using specific values, called arguments, in a particular order, or structure. Functions can be used to perform simple or complex calculations. Functions can become more powerful. They can take parameters, which mean that we can pass variables to a function for the function to work on. Functions can also be nested.

Common OBIEE Function Types

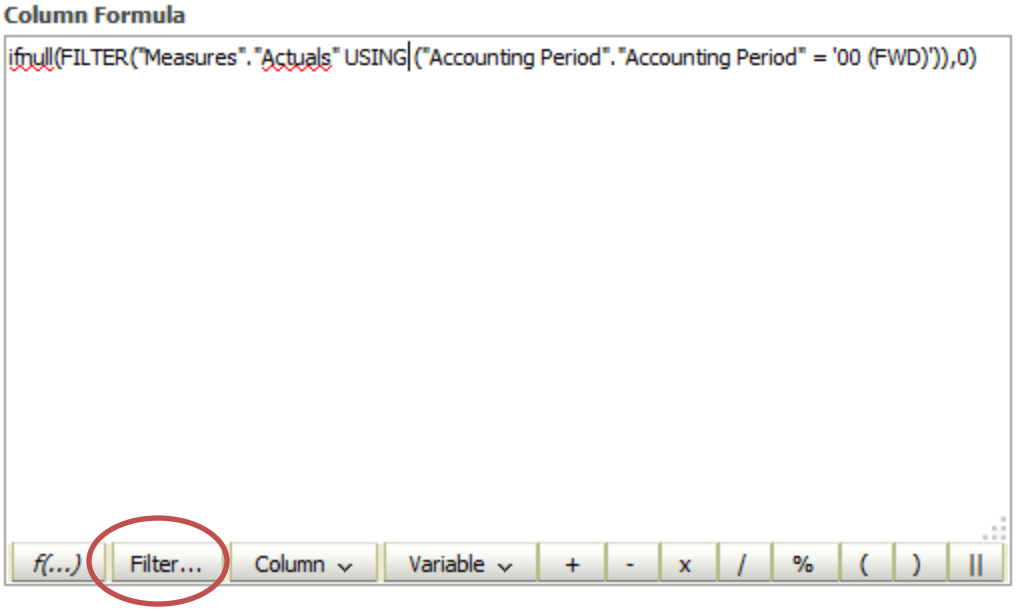
- Numeric Functions
- Character functions
- Date Functions
- Conversion Functions
- Grouping (aggregate) Functions
- Others



Syntax – double quotes around fields, single quotes around strings, commas between arguments, parenthesis enclosing arguments.

Here's a complicated column, using the filter function to limit the "Actuals" value by using the value of a repository variable, nested into another function of making sure it is set to zero if it is empty (null):

IFNULL(Filter("Measures"."Actuals" USING "Accounting Period"."Accounting Period (ID)" = VALUEOF("CURRENT_ACCOUNTING_PERIOD") - 1), 0.0)



MATH FUNCTIONS

Numeric functions are used to perform operations on numbers. They accept numeric values as input and return numeric values as output. Few of the Numeric functions are:

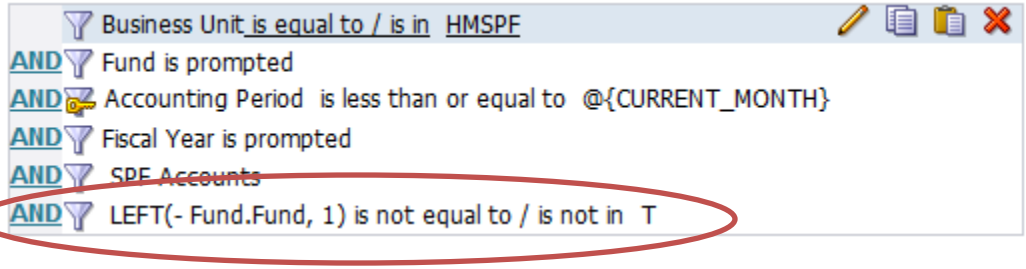
Function Name	Return Value	Examples	Return Value
ABS (x)	Absolute value of the number '*x*'	ABS (1) ABS (-1)	1 1
CEILING (x)	Integer value that is Greater than or equal to the number '*x*'	CEIL (2.83) CEIL (2.49) CEIL (-1.6)	3 3 -1
FLOOR (x)	Integer value that is Less than or equal to the number '*x*'	FLOOR (2.83) FLOOR (2.49) FLOOR (-1.6)	2 2 -2
TRUNCATE (x, y)	Truncates value of number '*x*' up to '*y*' decimal places	TRUNC (140.234, 2) TRUNC (-54, 1) TRUNC (5.7) TRUNC (142, -1)	140.23 54 5 140
ROUND (x, y)	Rounded off value of the number '*x*' up to the number '*y*' decimal places	ROUND (125.456, 1) ROUND (125.456, 0) ROUND (124.456, -1)	125.4 125 120

STRING FUNCTIONS

Character or text functions are used to manipulate text strings. They accept strings or characters as input and can return both character and number values as output.

Function Name	Return Value	Examples	Return Value
LOWER (*string_value*)	All the letters in *'string_value'* is converted to lowercase.	LOWER('Good Morning')	good morning
UPPER (*string_value*)	All the letters in *'string_value'* is converted to uppercase.	UPPER('Good Morning')	GOOD MORNING
TRIM (LEADING *string_value* FROM *trim_text*)	All occurrences of *'trim_text'* is removed from the left of *'string_value'*.	LTRIM ('Good Morning', 'Good)	Morning
TRIM (TRAILING *string_value* FROM *trim_text*)	All occurrences of *'trim_text'* is removed from the right of*'string_value'* .	RTRIM ('Good Morning', ' Morning')	Good
TRIM (BOTH *trim_text* FROM *string_value*)	All occurrences of *'trim_text'* from the left and right of *'string_value'* ,*'trim_text'* can also be only one character long .	TRIM ('o' FROM 'Good Morning')	Gd Mrning
SUBSTRING (*string_value* FROM *m* FOR *n*)	Returns *'n'* number of characters from*'string_value'* starting from the '*m'*position.	SUBSTR ('Good Morning', 6, 7)	Morning
LEFT(expr,integer n)	Returns n number of characters from the left side of a string	LEFT('Good Morning',1)	G
LENGTH (*string_value*)	Number of characters in *'string_value'*in returned.	LENGTH ('Good Morning')	12
string || *string*	Concatenates string values	vendor_city || ', ' || vendor_state || ' ' || vendor_postal_cd	Eureka, CA 95501

Example:



CALENDAR/DATE FUNCTIONS

Date Functions: These are functions that take values that are of datatype DATE as input and return values of datatypes DATE, except for the MONTHS_BETWEEN function, which returns a number as output.

Function Name	Return Value	Examples	Return Value
CURRENT_DATE CURRENT_TIME CURRENT_TIMESTAMP or NOW	Returns the systems current date, time, and date/timestamp.		
TIMESTAMPADD(*interval, n,timestamp*)	Adds a specific number of intervals (n) to a specified timestamp, returning a single timestamp. Intervals can be: SQL_TSI_SECOND,SQL_TSI_MINUTE, SQL_TSI_HOUR, SQL_TSI_DAY, SQL_TSI_WEEK, SQL_TSI_MONTH, SQL_TSI_QUARTER, SQL_TSI_YEAR	TIMESTAMPADD(sql_tsi_month, 1,"- Accounting Date"."Accounting Date")	01/01/2013 would return 02/01/2013
TIMESTAMPDIFF(*interval, ts1, ts2*)	Returns the number of intervals between timestamps ts1 and ts2.	TIMESTAMPDIFF(SQL_TSI_DAY,"- Event Dates"."Hire Date",CURRENT_DATE)	The number of days from the hire date to today.

Example: How many days elapsed between Job Date and Today?

Column Formula

```
TIMESTAMPDIFF(SQL_TSI_DAY, "Date"."Job Date",CURRENT_DATE)
```

Example – NOW function:

General Fund Expenditures

Fund by Object Group

Fund	Object Group	Original Budget	Revised Budget	Actuals	Encumbrances	Balance Remaining	Percent of Budget Spent
HM500 - OPERATING FUND	601 - Regular Salaries and Wages	618,582.00	620,406.00	567,133.72	0.00	53,272.28	91.41
	603 - Benefits Group	253,639.00	262,672.00	240,468.99	0.00	22,203.01	91.55
	604 - Communications	2,971.00	3,371.00	3,104.93	0.00	266.07	92.11
	606 - Travel	40,000.00	54,284.00	41,443.69	15,474.82	(2,634.51)	104.85
	608 - Library Acquisitions	0.00	0.00	82.50	0.00	(82.50)	
	613 - Contractual Services Group	2,000.00	2,000.00	2,000.00	0.00	0.00	100.00
	616 - Information Technology Costs	0.00	11,341.00	11,271.22	0.00	69.78	99.38
	619 - Equipment Group	0.00	18,800.00	6,787.98	0.00	12,012.02	36.11
	660 - Misc. Operating Expenses	24,582.00	14,859.00	11,927.75	794.00	2,137.25	85.62
HM500 - OPERATING FUND Total		**941,774.00**	**987,733.00**	**884,220.78**	**16,268.82**	**87,243.40**	**91.17**
Grand Total		**941,774.00**	**987,733.00**	**884,220.78**	**16,268.82**	**87,243.40**	**91.17**

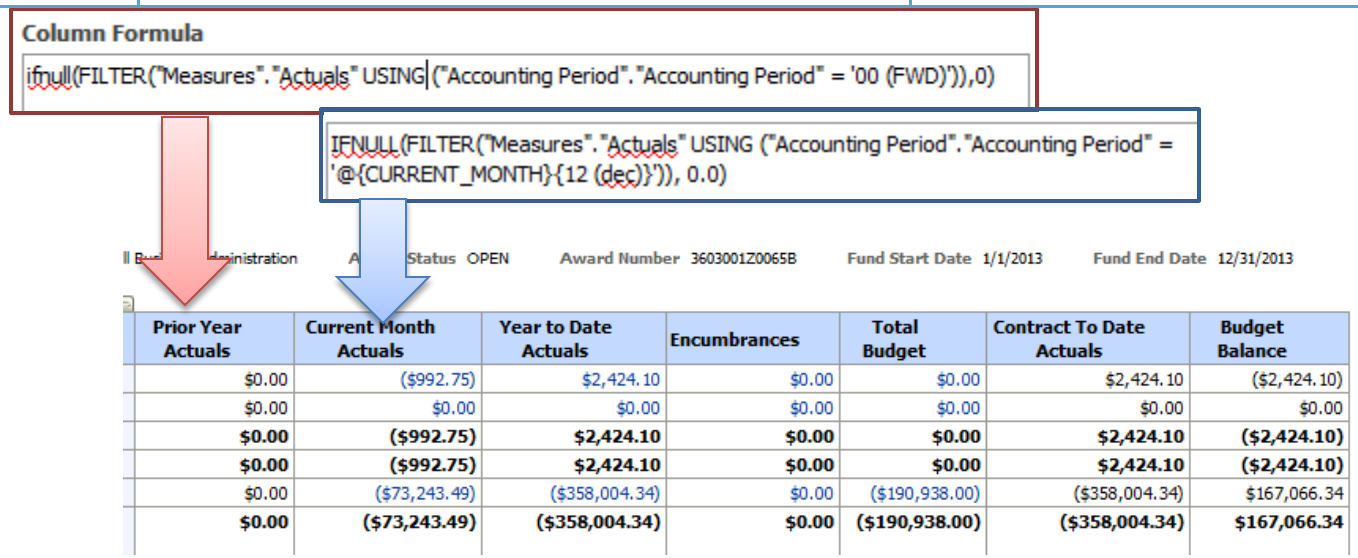
Run Date: 6/27/2013 2:40:32 PM

CONVERSION FUNCTIONS

Conversion Functions: These are functions that help us to convert a value in one form to another form. For Ex: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE.
Few of the conversion functions available in oracle are:

Function Name	Return Value	Examples	Return Value
CAST (*x* AS *y*)	This function changes the data type of an expression or a null literal to another data type. Most commonly used datatype values for 'y' are: CHAR, VARCHAR, INTEGER, DOUBLE PRECISION, DATE, TIME, TIMESTAMP NOTE: If you use the CHAR datatype, you can add a size parameter. If omitted, a default of 30 is used. If you use the VARCHAR datatype, you MUST provide a size parameter.	CAST("- Event Dates"."Hire Date" AS CHAR)	09/13/2011 converts to '13-SEP-11'
IFNULL (*x, y*)	If *'x'* is NULL, replace it with *'y'*. *'x'* and *'y'* must be of the same datatype. If 'y' is a string value, it must be enclosed in single quotes.	IFNULL("- Demographics"."Ethnic Group",'Not Specified')	
VALUEOF(*expr*)	Use the VALUEOF function to reference the value of an Oracle BI repository variable.	VALUEOF(current_fiscal_year)	2012

Example:



Prior Year Actuals	Current Month Actuals	Year to Date Actuals	Encumbrances	Total Budget	Contract To Date Actuals	Budget Balance
$0.00	($992.75)	$2,424.10	$0.00	$0.00	$2,424.10	($2,424.10)
$0.00	$0.00	$0.00	$0.00	$0.00	$0.00	$0.00
$0.00	**($992.75)**	**$2,424.10**	**$0.00**	**$0.00**	**$2,424.10**	**($2,424.10)**
$0.00	**($992.75)**	**$2,424.10**	**$0.00**	**$0.00**	**$2,424.10**	**($2,424.10)**
$0.00	($73,243.49)	($358,004.34)	$0.00	($190,938.00)	($358,004.34)	$167,066.34
$0.00	**($73,243.49)**	**($358,004.34)**	**$0.00**	**($190,938.00)**	**($358,004.34)**	**$167,066.34**

AGGREGATE FUNCTIONS

Aggregate functions return a single result row based on groups of rows, rather than on single rows.

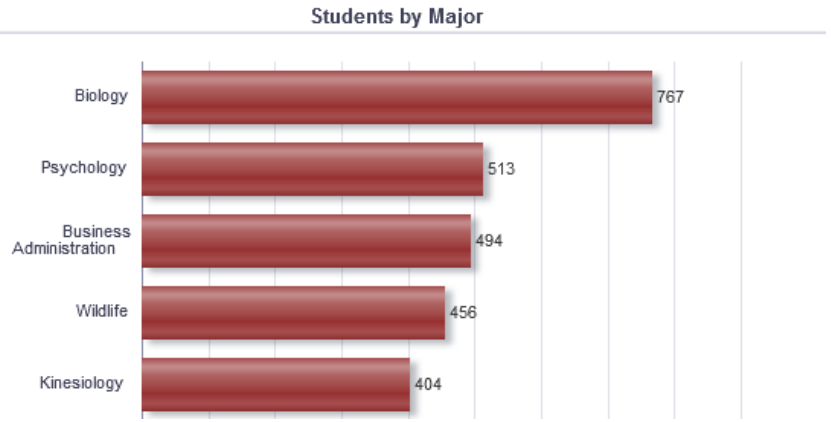
Function Name	Return Value	Examples	Return Value
AVG(*expr*)	Returns the average of the values in a set of rows	AVG(endowment_unit_value)	
COUNT(*expr*)	Returns the number of rows having a non-null value in the set	COUNT(DISTINCT univ_id_no)	
COUNT(DISTINCT *expr*)	Adds distinct processing to the Count function		
COUNT(*)	Counts the number of rows		
MAX(*expr*)	Returns the largest value from a set of rows	MAX(tub_last_update_dt)	
MIN(*expr*)	Returns the smallest value from a set of rows	MIN(fringe_assessment_rate)	
SUM(*expr*)	Adds the value for all rows in the set	SUM(pcard_transaction_distr_amt)	
TOPN(*expr*, *integer*) BOTTOMN(*expr*, *integer)*	Ranks the highest (or lowest) n values of the expression argument from 1 to n, corresponding to the highest (or lowest) numerical value	TOPN("Measures"."Actuals",10)	Lists the top 10 by total actual expense (by dept for example)
RANK(*expr*)	Calculates the rank of the expression for all values in the set	RANK("Measures"."Actuals")	Shows rank of all actuals in analysis

Example: Enrollment by Major

Column Formula

```
Rank("Fact Term Enrollment"."Term Enrollment Count")
```

		Term Enrollment Count
		Fall Semester 2013
Rank Count	Study Field Desc	
1	Biology	767
2	Psychology	513
3	Business Administration	494
4	Wildlife	456
5	Kinesiology	404



VARIABLES

There are 3 types of variables used in OBI: Repository, Session, and Presentation. Repository and session variables are also known as *Server Variables.* Each is described below.

Repository Variables

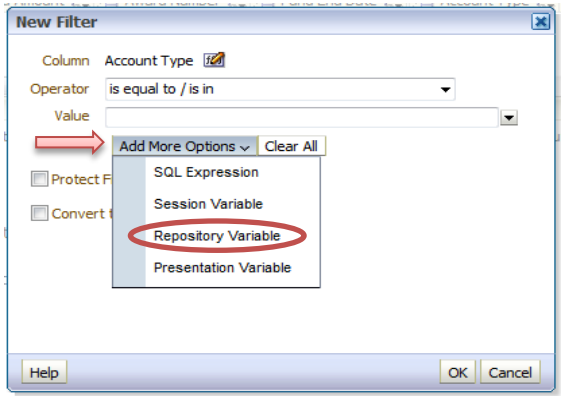
A repository variable has a single value at any point in time. It is created and initialized (set) in the Repository and can be used in prompts, filters and in the analysis itself. There are two types of Repository Variables:

- STATIC: The value of a Static Repository variable never changes. You can use these to define things, like "Prime Time" being between the hours of 7 p.m. and 11 p.m.

- DYNAMIC: The value of a Dynamic Repository variable is refreshed by data returned from queries. These values can usually change from day to day, or hour to hour, depending on the interval that the data is refreshed.

Referencing a Repository Variable

To use a repository variable in a filter, select the column to filter, and then instead of selecting a value, choose 'Add More Options'. You will see a dropdown to select the type of variable you want to use.

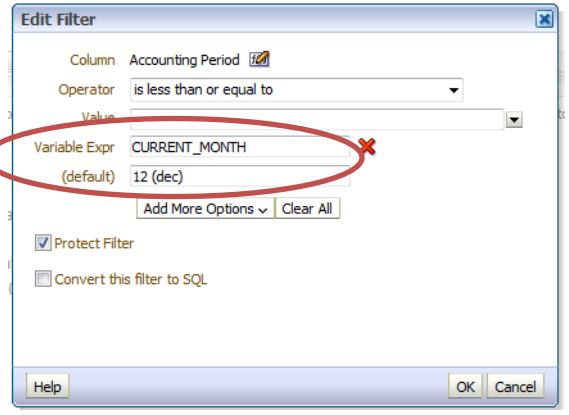


When you select Repository Variable, you can then enter the variable name and the default value you would like to use. Be careful setting the default – you need to enter it *exactly* right, as there is no list to select from.

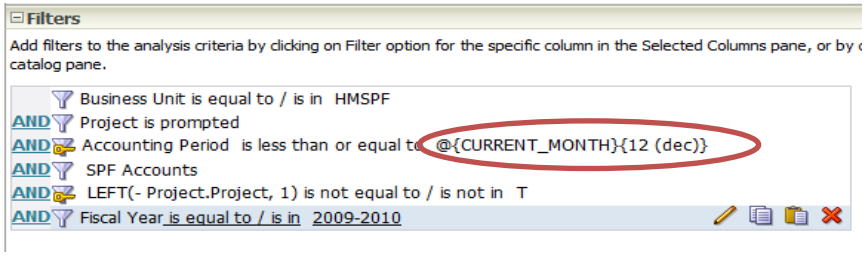
When you click OK to set the filter, the value will be set as follows:

```
@{VariableName}{<default>}
```

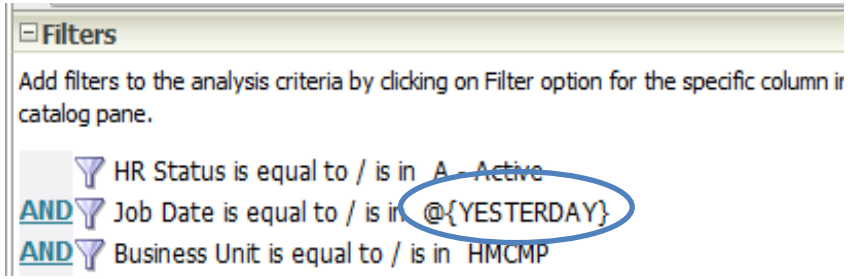
- VariableName: a variable name that is not a reserved name (see list at end of section)
- Default: (optional) A default value for the variable



NOTICE that the variable name and default portions are in squiggly brackets, while, and everything is preceeded by an "@" sign.



The following are the Dynamic Repository Variables we have currently defined in the Data Warehouse OBI RPD (to the right) and an example of their use (below):



Variable Name	Format	Example
CURRENT_ACCOUNTING_PERIOD	decimal	6.0
CURRENT_FISCAL_YEAR	year (decimal)	2012.0
CURRENT_YEAR	year (text)	2013
CURRENTDATE	timestamp	2013-06-28 09:24:24
PRIOR_FISCAL_YEAR	year (decimal)	2011.0
DATE_PRIOR_30_DAYS	timestamp	2013-05-29 09:24:24
DATE_PRIOR_60_DAYS	timestamp	2013-04-29 09:24:24
DATE_PRIOR_90_DAYS	timestamp	2013-03-30 09:24:24
DATE_PRIOR_180_DAYS	timestamp	2012-12-30 09:24:24
DATE_PRIOR_ONE_YEAR	timestamp	2012-06-28 09:24:24
DATE_PRIOR_TWO_YEARS	timestamp	2011-06-28 09:24:24
DATE_PRIOR_FIVE_YEARS	timestamp	2008-06-28 09:24:24
ONE_WEEK_PRIOR	timestamp	2013-06-21 09:24:24
ONE_YEAR_PRIOR	year (text)	2012
TWO_YEAR_PRIOR	year (text)	2011
THREE_YEAR_PRIOR	year (text)	2010
FOUR_YEAR_PRIOR	year (text)	2009
FIVE_YEAR_PRIOR	year (text)	2008
SIX_YEAR_PRIOR	year (text)	2007
SEVEN_YEAR_PRIOR	year (text)	2006
YESTERDAY	timestamp	2013-06-27 09:24:24

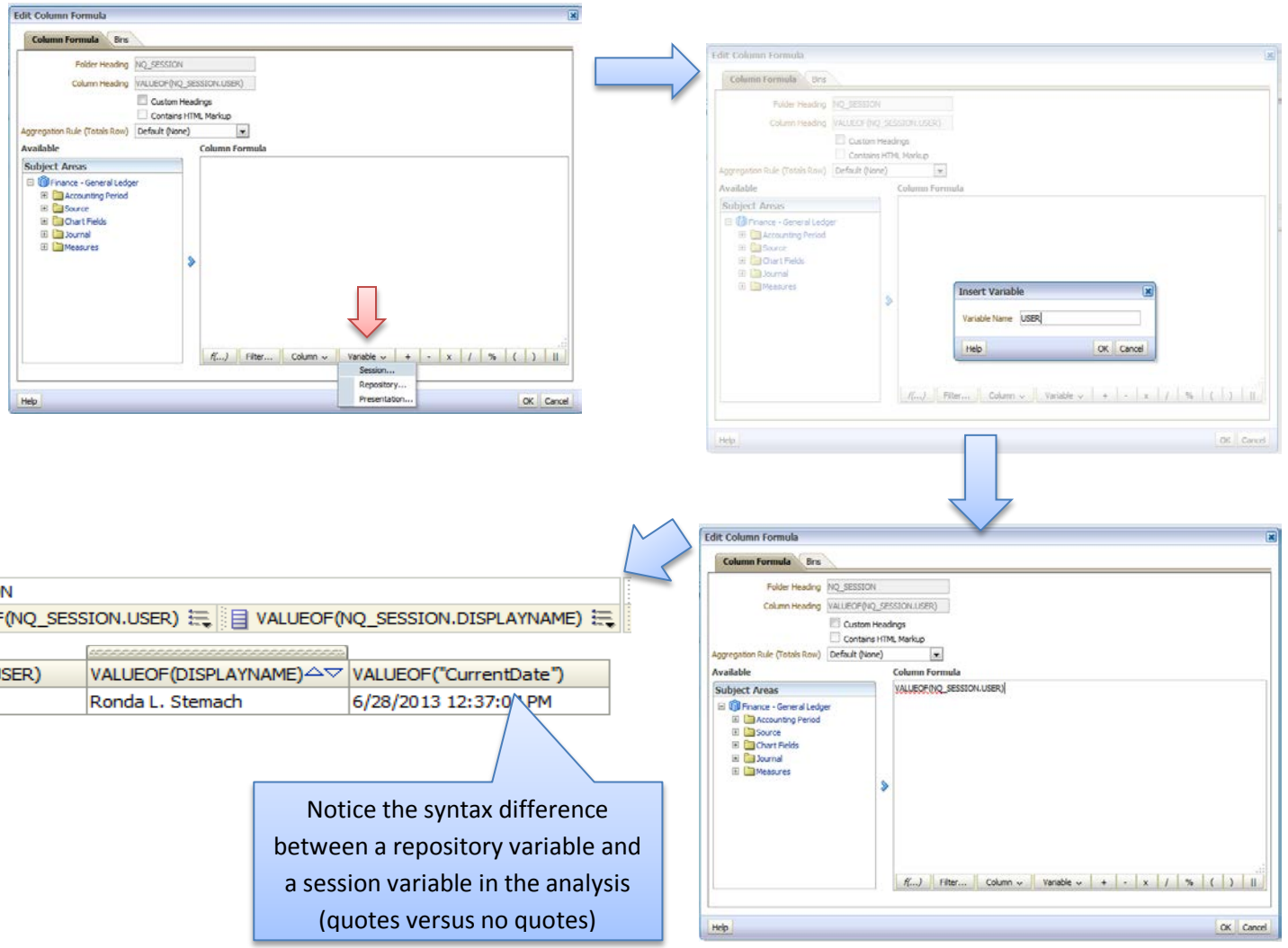
Session Variables

Session variables are user-based, and are initialized (set) when a user begins a session. There are two types: System and Nonsystem.

- SYSTEM: These variables have a specific purpose in OBI, and have a "NQ_SESSION" prefix which automatically attaches to it. The most commonly used are ones to identify the user, both by their ldap username and also their display name. .

- NONSYSTEM: These variables have a customized definition in OBI, and also have a "NQ_SESSION" prefix which automatically attaches to it. An example of a nonsystem session variable would be something related to the user, such as a sales region, so when the user signs in the variable would be populated with the region.

System Variable Example:



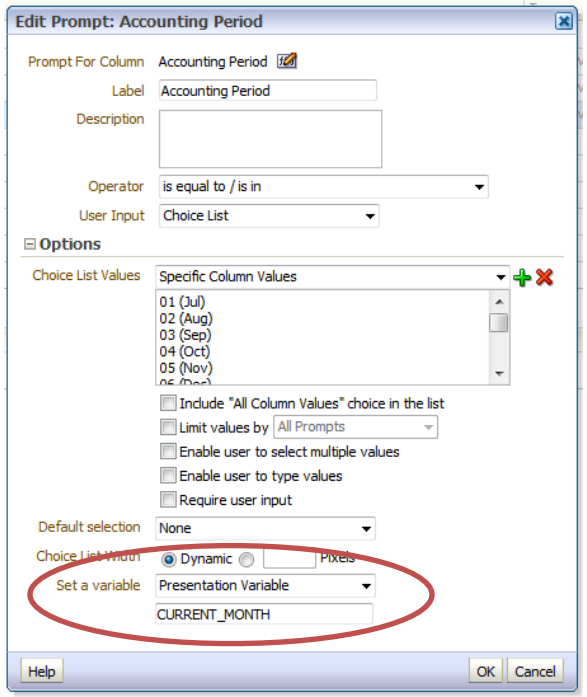
Presentation Variables

Presentation variables are user defined, and can be references in various areas, such as the dashboard, the analysis or the prompt. If set by a dashboard prompt, it will take the data type of the presentation column.

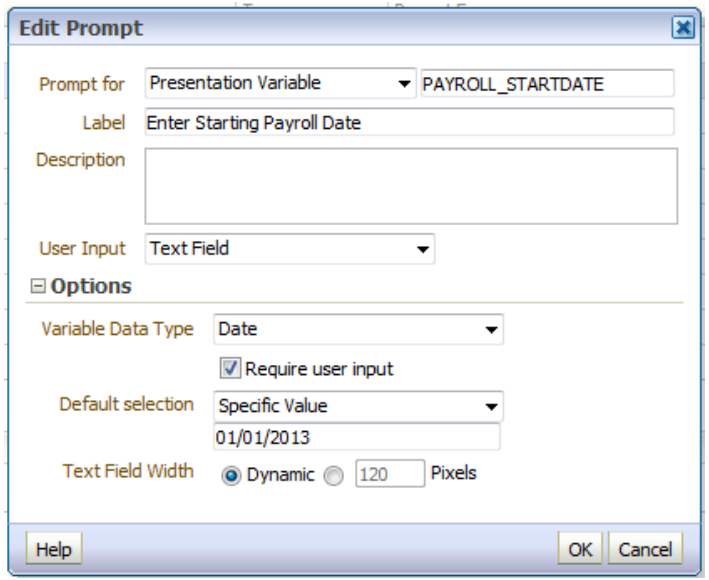
How to set a Presentation Variable

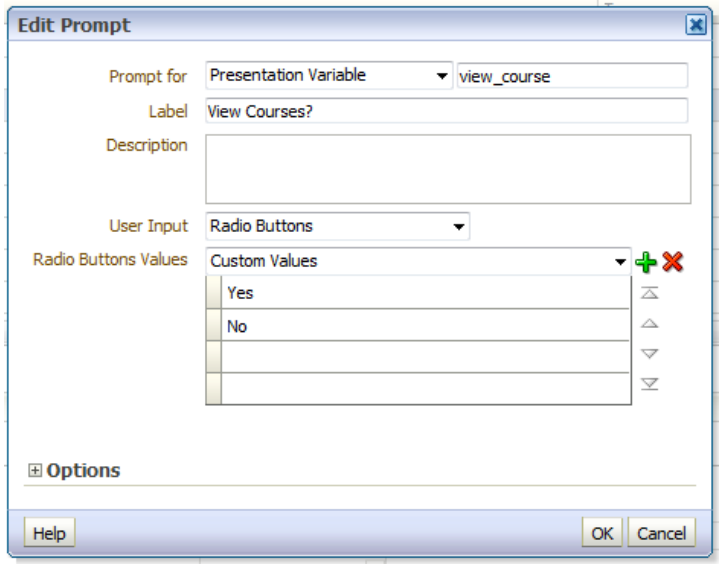
To set a Presentation Variable from a Prompt, use the 'Set a Variable' section at the bottom of the Edit Prompt dialog box.

In this section, you can set two types of variables: Presentation and Request. A Presentation variable is the most commonly used; however, you have the ability to *temporarily override* the value of a session variable by using the option to set a Request Variable. The value of the repository session variable is not changed. The new value is only applicable for the current analysis, or "request".



You can also create a prompt that is not based on an existing column, but instead prompts the user to enter the value directly.





How to reference a Presentation Variable

The syntax for referencing presentation variables is as follows:

```
@{VariableName}{<default>}[format]
```

- VariableName: a variable name that is not a reserved name (see list at end of section)
- Default: (optional) A default value for the variable
- Format: (optional) A format mask based on the data type of the variable.

NOTICE that the variable name and default portions are in squiggly brackets, while the format is a square bracket, and everything is preceeded by an "@" sign.

Some examples in use: