QlikView

QLIKVIEW DATE FIELDS

QlikView Technical Brief

May 2012, HIC

qlikview.com

Introduction

In most cases, you will have no problems loading dates in QlikView. The software has an intelligent algorithm to recognize dates independently of which region you are in. It usually just works and you do not need to think about it. However, in some cases dates are not properly recognized and then you need to add some code in the script to make it work.

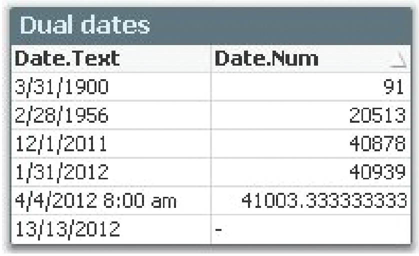
However, many of the questions on QlikCommunity concern dates and how to use date functions in QlikView. Although the basics for dates are fairly simple, there are many misunderstandings. This technical brief is an attempt to give some background and some suggestions. Those of you who are experienced QlikView users may not find anything new in this document, but others most likely will.

Data types

First – there are no data types in QlikView. Instead QlikView uses a dual data storage for all field values: every field value is represented by a string and – if applicable – a number. If a field value can be interpreted as a number (or a date) then a number is stored in the numeric part of the dual storage, while the display string is stored in the textual part of the dual storage.

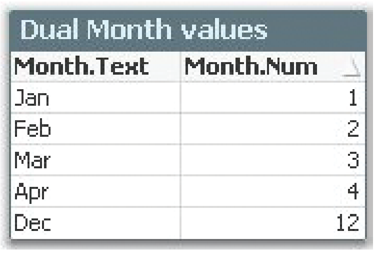
The serial numbers used for dates are the same as in Excel: the number of days passed since the 30th December 1899 using the Gregorian calendar. The integer part of the serial number is the date and the fractional part is the time of the day. For example, if the 31st of January 2012 at 6 o'clock in the morning is loaded with US date format, then the number 40939.25 is stored together with the string '1/31/2012 6:00 am'.

This way dates, week days and month names can have textual names or arbitrary formats and still be numerically sorted. The fields can also be used in numerical calculations and comparisons and as continuous axes in graphs. Numerical functions always use the numeric part of the dual field and string functions always use the string part.



Dual dates

Date.Text	Date.Num
3/31/1900	91
2/28/1956	20513
12/1/2011	40878
1/31/2012	40939
4/4/2012 8:00 am	41003.333333333
13/13/2012	-



Dual Month values

Month.Text	Month.Num
Jan	1
Feb	2
Mar	3
Apr	4
Dec	12

Interpretation and formatting

There are two important types of QlikView functions that deal with time and dates: Interpretation functions and Formatting functions. The interpretation functions – e.g. Date#() and TimeStamp#() - are string-to-number conversions, i.e. the input is a string that contains a date and the function creates a correct date serial number. The output is a dual field, i.e. both string and number.

The formatting functions – e.g. Date() and TimeStamp() – are the opposite: they are number-to-string conversions, i.e. the input is a date serial number and the function creates a string with the properly formatted date. Also here the output is a dual field, i.e. both string and number.

You rarely need to use the interpretation or formatting functions. Normally it just works anyway. The reason for this, is that if there is no interpretation function explicitly used, QlikView tries with the date format specified in the format variables, e.g. it uses what it finds in the "Set DateFormat = …" statement in the beginning of the script, which usually is just what you need.

Here are some suggestions of how to work with dates and times if your fields aren't interpreted correctly:

TIP 1: USE INTERPRETATION FUNCTIONS

If the date isn't automatically recognized, you may need to use an interpretation function to interpret it:

```
Date#( DateField, 'M/D/YY') as Date
```

Make sure you really use an interpretation function and not a formatting function – it should have a hash sign "#" in it.

Also, you should check that QlikView really has interpreted the dates as numbers: Either implicitly by checking that the dates are right-aligned in the list box, or explicitly by formatting the dates as numbers (Properties – Numbers) and verifying that the numbers displayed have values around 40000 (for dates in present time).

The format code used is the same as in Excel. Note that the letters in it are case sensitive. "M" means months and "m" means minutes.

The string does not have to be a complete date. A partial date can also be interpreted. If you for instance have a field with month names only in it, you can convert the textual month names to dual months using:

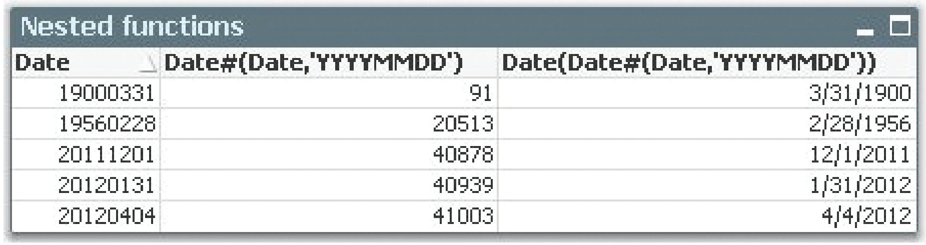
```
Month( Date#( Month,'MMM') ) as Month
```

TIP 2: NEST FUNCTIONS

It is often practical to nest an interpretation function inside a formatting function, e.g.

```
Date( Date#( DateField, 'M/D/YY'), 'YYYY-MM-DD') as Date
```

The inner function ensures that the input text is interpreted correctly – so that a serial number representing the date is created. The outer function ensures that the serial number is displayed in a correct date format.



Nested functions

Date	Date#(Date,'YYYYMMDD')	Date(Date#(Date,'YYYYMMDD'))
19000331	91	3/31/1900
19560228	20513	2/28/1956
20111201	40878	12/1/2011
20120131	40939	1/31/2012
20120404	41003	4/4/2012

Example:

You have a date field where the dates are stored as '20120131', i.e. no slashes or dashes between the day, month and year. QlikView will incorrectly interpret this as an integer number with a value of slightly more than 20 millions. I say 'incorrectly' since it assigns values that are other than the correct date serial numbers, making comparisons between dates impossible. If you for instance want to calculate the number of days between 20120131 and 20120201, you would get 70, when it obviously should be 1.

One correct way to load it could be

```
Date( Date#( DateField, 'YYYYMMDD'), 'M/D/YYYY') as Date
```

where the inner function contains the actual date format and the outer contains the preferred date format. Also, this way the dates get numeric values of around 40000 and can be correctly compared to other dates and used in calculations.

TIP 3: USE THE MAKEDATE FUNCTION

If your date is stored in several fields, e.g. one field for year, a second for month and a third for day, you should use the MakeDate function to create a proper date serial number for the specified day:

```
MakeDate( Year, Month, Day ) as Date
```

TIP 4: USE THE ROUNDING FUNCTIONS

The date field in the source data is often not just a date, but instead a timestamp corresponding to a specific time during the day. The date serial number will then not be an integer. For instance, the time 6 pm 1/1/2012 corresponds to the date serial number 40909.75.

In such a case it is not enough to use the date function to remove hours and minutes from the formatting. Though formatting the timestamp as a date will hide the time from being displayed, the fractional part of the serial number will still be there and the field may give incorrect results in comparisons.

Instead a rounding function must be used to make the additional 0.75 from the numeric value disappear, i.e.:

```
Date( Floor( Date ), 'YYYY-MM-DD') as Date
```

Another case where a rounding function is good is when the date is a key field linking two tables. If the field value is a timestamp where you have a time of the day other than midnight – 00:00:00 – then this value will not link to a date in another table even if you have formatted it as a date: The string part of the dual format is not used as key if there is a numeric value. The numeric value is always used as key. Hence it is not enough that two numeric values are formatted exactly the same. If you want to use a date as a key, you should use the integer part of the timestamp and omit the information about time of the day.

Example:

You have a timestamp, e.g. ‘2012-01-28 08:32:45’ in your transaction table and you want to link this to a master calendar table containing dates. One correct way to load this key could be

```
Date( Floor( Timestamp#( Created, 'YYYY-MM-DD hh:mm:ss')), 'M/D/YYYY')
as CreatedDate
```

In addition to the key CreatedDate, other fields could also be created to show the fractional part of the timestamp

```
Timestamp#( Created, 'YYYY-MM-DD hh:mm:ss') as CreatedTimestamp,

Time( Frac( Timestamp#( Created, 'YYYY-MM-DD hh:mm:ss')), 'hh:mm:ss')
as CreatedTime,
```

Also note that the function Frac() is used to remove the integer part of the number for the field that only contains the time information.

TIP 5: ALWAYS USE THE NUMERIC VALUE IN VARIABLES

Fields are dual, but variables are not. This means that whenever you want to store a date in a variable and use it later for e.g. a numeric comparison in a where clause, it is easier if the variable is numeric instead of a string containing a date format. Hence, use the following construction:

```
Let vToday = Num( Today() ) ;
Let vAMonthAgo = Num( AddMonths( Today(),  – 1 ) ;
```

Subsequent use of the variable is then straightforward, e.g.:

```
… Where DeliveryDate  > $(#vAMonthAgo) and DeliveryDate <= $(#vToday);
```

Note the hash sign in the dollar expansions: it forces an expansion using decimal point. This is helpful for users with decimal comma (and not decimal point) in the national settings.

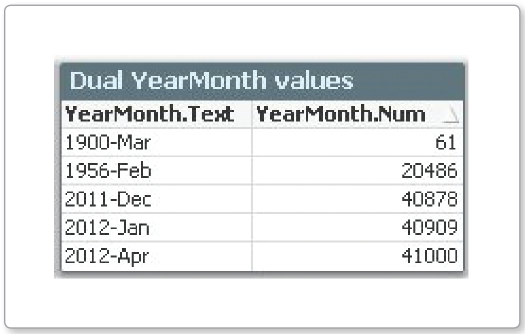
TIP 6: USE COMBINATION FIELDS, E.G. YEAR AND MONTH AS ONE FIELD

Date numbers should be used for year-month fields and other similar situations:

```
Date( MonthStart( DateField ), 'YYYY-MMM') as YearMonth
```

Here the MonthStart() function returns the date number of the beginning of the month and the Date() function is used to hide the day of the month.

There are other similar useful functions like WeekStart, QuarterStart and YearStart.



Dual YearMonth values

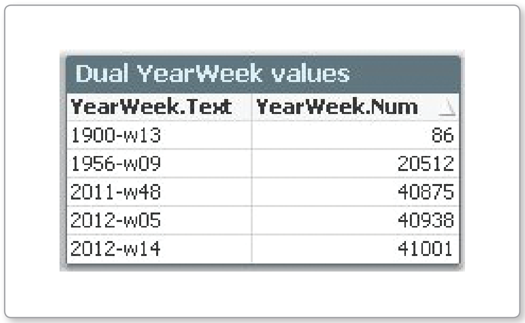
YearMonth.Text	YearMonth.Num
1900-Mar	61
1956-Feb	20486
2011-Dec	40878
2012-Jan	40909
2012-Apr	41000

TIP 7: USE THE DUAL FUNCTION

The Dual function can often be used to solve trickier problems. The Dual function lets you specify both a numeric value as well as which text to associate with this value.

For instance, if you want to use the week number, but sorted correctly also over the change of the year, then you should use the date number as numeric value but display the week number, optionally together with the year:

```
Dual( Week( Date ), WeekStart( Date ) ) as YearWeek
```



Dual YearWeek values

YearWeek.Text	YearWeek.Num
1900-w13	86
1956-w09	20512
2011-w48	40875
2012-w05	40938
2012-w14	41001

```
Dual( Year( Date ) & '-w' & Week( Date ), WeekStart( Date ) ) as YearWeek
```

Or if you want to create fiscal months:

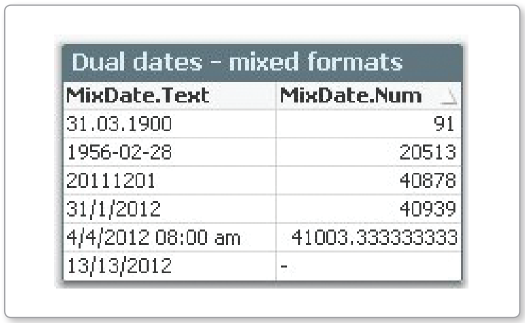
```
Dual( Month(Date), Mod(Month(Date)-$(vFirstMonthOfFiscalYear),12)+1)
```

TIP 8: FIELDS WITH MIXED DATE FORMATS

If you have mixed date format in one field or you have data from different sources using different formats, you can use the Alt() function to define several possible date formats:

```
Alt(    Timestamp#(MixDate,'M/D/YYYY h:mm tt'),
        Date#(MixDate,'M/D/YYYY'),
        Date#(MixDate,'D/M/YYYY'),
        Date#(MixDate,'YYYYMMDD'),
        Date#(MixDate,'DD.MM.YYYY'),
        Date#(MixDate'YYYY-MM-DD')
        ) as MixDate
```

The order of the interpretation functions also defines the priority when several formats are possible for one specific field value, e.g. 8/4/2012, which in the United States means 4th of August, but in the United Kingdom means 8th of April.



Dual dates - mixed formats

MixDate.Text	MixDate.Num
31.03.1900	91
1956-02-28	20513
20111201	40878
31/1/2012	40939
4/4/2012 08:00 am	41003.333333333
13/13/2012	-

Summary

Always convert anything date or time like to a proper date serial number and use this as the number in the dual format. The text part of the dual format is a display topic; it is up to you to decide what to show. Good Luck!

Inter gravissimas – The Gregorian calendar

… or some interesting but useless facts:

The QlikView serial numbers are well defined and behaving correctly according to a generalized Gregorian calendar from Jan 1st, 0 AD and roughly 2 billion years forward. As a comparison, you can note that the Excel date functions are defined from March 1 1900. If you enter dates before this, Excel will incorrectly assume that 1900 was a leap year.

If you are a calendar aficionado, you might argue that there was no year 0; that the year 1 AD was preceded by the year 1 BCE without an intervening year 0. This is correct – historians have never included a year zero. This system was designed by the monk Dionysius Exiguus in the 6th century, when the existence of zero as a number was not known in Europe and the Julian calendar was used.

The Gregorian calendar was introduced much later, in 1582 by Pope Gregory XIII. The papal bull did not say anything about how the new calendar should be applied backwards in time. As a consequence, when the new calendar was extended backwards, the old logic from the 6th century was used. This is today called the proleptic Gregorian calendar. It has no year zero and the common AD/CE or BC/BCE notation is used.

But there is a second generalization of the Gregorian calendar: the astronomical year numbering, with a separate year zero. Very early, astronomers used a separate year for year 0 (Christi) in the astronomical tables, as opposed to the years before (Ante Christum) and after (Post Christum), e.g. Johannes Kepler (1627, the Rudolphine Tables) and Philippe de la Hire (1702, Tabulæ Astronomicæ). However, it was not until 1740 that Jacques Cassini introduced the number 0 to mark this year in his Tables Astronomiques.

The astronomical year numbering is now an ISO standard (ISO8601) and is used worldwide in all scientific contexts. Hence, year -0001 (ISO 8601) is the same as year 2 BCE (proleptic Gregorian). QlikView follows the ISO8601 standard.

Unfortunately many QlikView date functions do not work properly for the years before 1 AD. However, it is still possible to use QlikView serial numbers for this period. The things to be aware of are:

- Several functions do not work properly, e.g. Date(), Date#() and MakeDate(). Hence, formatting the date will produce something incorrect.
- If you want to use the proleptic Gregorian calendar, you can use the following as year function:
  Num(if(Year(Date)>0, Year(Date), Year(Date)-1),'#0 AD;#0 BCE') as Year
- You can generate the dates using something similar to e.g.
  Load Dual(Num(Year(Date),'0000') &'-' & Date(Date,'MM-DD'),Date) as Date;
  Load MakeDate(1)-recno() as Date autogenerate 100000