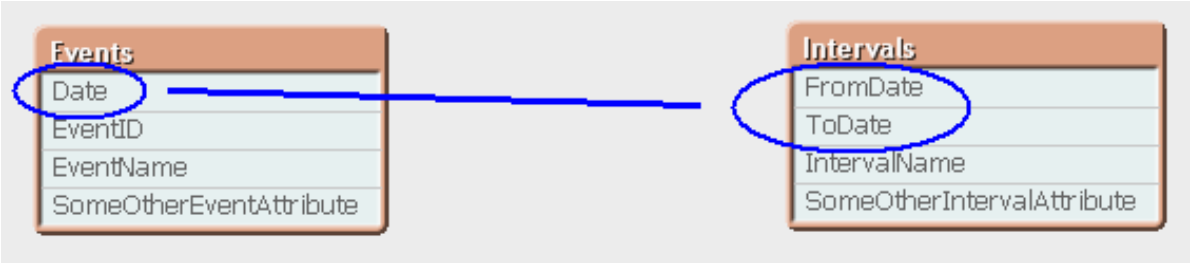
QlikView Design Blog : IntervalMatch

Posted by Henric Cronström Apr 4, 2013

A common problem in business intelligence is when you want to link a number to a range. It could be that you have a date in one table and an interval – a “From” date and a “To” date – in another table, and you want to link the two tables. In SQL, you would probably join them using a BETWEEN clause in the comparison.



But how do you solve this in QlikView, where you should avoid joins?

The answer is to use IntervalMatch.

IntervalMatch is a prefix that can be put in front of either a Load or a SELECT statement. The Load/SELECT statement needs to contain two fields only: the “From” and the “To” fields defining the intervals. The IntervalMatch will generate all the combinations between the loaded intervals and a previously loaded numeric field.

Typically, you would first load the table with the individual numbers (The Events), then the table with the Intervals, and finally an intervalmatch that creates a third table that bridges the two first tables.

Events:

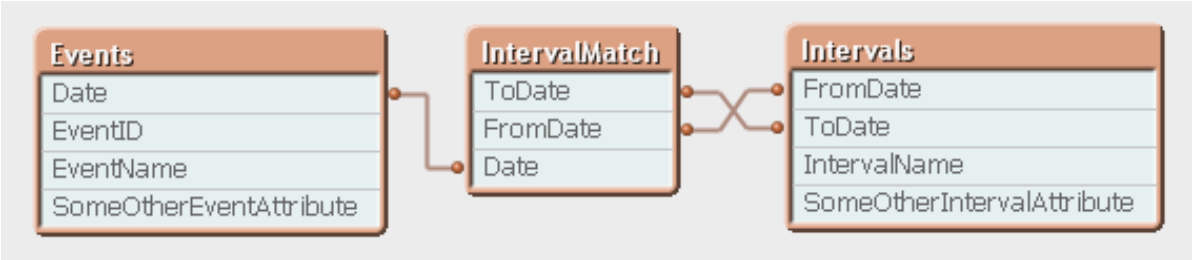
```
Load * From Events;
```

Intervals:

```
Load * From Intervals;
```

IntervalMatch:

```
IntervalMatch (Date)
Load FromDate, ToDate resident Intervals;
```



The resulting data model contains three tables:

1. The Events table that contains exactly one record per event.
2. The Intervals table that contains exactly one record per interval.
3. The IntervalMatch table that contains exactly one record per combination of event and interval, and that links the two previous tables.

Note that this means that an event may belong to several intervals, if the intervals are overlapping. And an interval can of course have several events belonging to it.

This data model is optimal, in the sense that it is normalized and compact. All QlikView calculations operating on these tables e.g. *Count(EventID)* will work and will be evaluated correctly. This means that it is ***not*** necessary to join the intervalmatch table onto one of the original tables. Joining it onto another table may even cause QlikView to calculate aggregations incorrectly, since the join can change the number of records in a table.

Further, the data model contains a composite key (the FromDate and ToDate fields) which will manifest itself as a QlikView synthetic key. But have no fear. This synthetic key ***should*** be there; not only is it correct, but it is also optimal given the data model. You do ***not*** need to remove it.

IntervalMatch can also be used with an additional key between the tables – i.e. when you have Slowly Changing Dimensions. But more about that in a later post.

HIC

290 Views Tags: intervalmatch, slowly_changing_dimension, intervals, prefix

Apr 4, 2013 10:08 AM paulyeo11

nice post , best example interval match is jennell P&L , which i like the best.

Apr 4, 2013 12:16 PM amirvas

Can also add additional dimension in cases where Employees change departments and you want to find the original person responsible for creating the record and/or sales etc. so that credit applies accordingly

I tend to LEFT JOIN the intervaltable into the fact table which removes the valid synthetic key just to keep things simple

Apr 4, 2013 5:51 PM Henric Cronström amirvas in response to

Your case is a a Slowly Changing Dimension. A record in the fact table has an EmployeeID and a Date. To find the department to which the employee belongs, you need both keys. An employee can (probably) only belong to one department at a time, and in such a case a Left Join can be used: QlikView will still calculate the aggregations correctly since the number of records in the fact table won't change.

However, a left join will add columns to the fact table - which will increase the memory usage. A solution without a left join could probably use less memory.

HIC

Apr 4, 2013 5:55 PM amirvas Henric Cronström in response to

Very true.

Apr 4, 2013 10:22 PM Trey Bayne

Saying that a synthetic key is ok here is just like saying that allowing QV to auto-concatenate is ok. Most of the time, QV is doing exactly what it should. The problem is that QV obscures what it does in synthetic keys. Doing a left join to the fact table guarantees what you are getting.