QlikView

CONCATENATE AND LINK TABLES

Use cases scenarios

QlikView Technical Brief

March 2013

QlikView

Contents

QlikView

About

QlikTech is a leader in Business Discovery — user-driven Business Intelligence (BI). Our QlikView Business Discovery platform is what's next in business software. Today's business leaders want to enable users at all levels of the organization to leverage data to drive innovative decisions that push the business forward. QlikView puts those business users in control of exploring and exploiting their data without limits by delivering these capabilities:

Insight Everywhere — Business Discovery is a whole new way of doing things that puts the business user in control. Unlike traditional BI, where just a few people are involved in insight creation, Business Discovery enables everyone to create insight.

QlikView

Introduction

Link Tables and concatenation of tables are used to resolve data knots like synthetic keys and/or circular references which normally occur when dealing with more than one fact table.

It´s not easy to find a general rule to apply on every possible situation about when you should Concatenate rather than joining tables by a Link Table so today we will try to put some light on it by highlighting the main differences.

QlikView

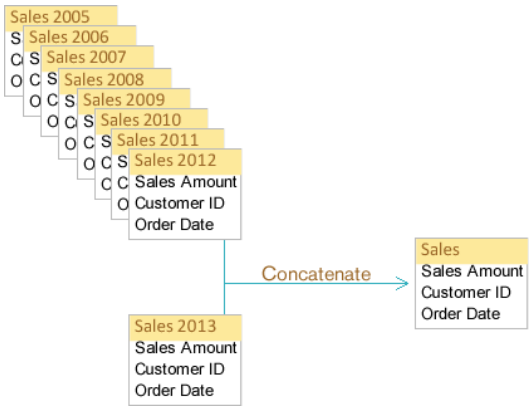
Concatenate

"This statement forces concatenation with an existing named table or the latest previously created Logical Table. A concatenation is in principle the same as the SQL UNION statement, but with two differences: first that Concatenate prefix can be used no matter if the tables have identical field names or not; and secondly that no removals of identical records are made"

Source: QlikView help

Most basic example of Concatenate usage could be when you need to merge two or more **tables that have identical structures**, let say you have a historical data warehouse with sales from 2005 to 2012 and then another table with 2013 sales live in you transactional database.

To create a visualization of sales trends over the years you will want to have everything normalized in one fact table, Sales table, with the data from 2005 to 2013.



QlikView's Concatenate function will also let you to force concatenation when these tables have not the same structures. This is pretty convenient when you need to merge data coming from different systems or databases.

Tip: If the tables are identical QlikView will concatenate them automatically.

Let examine another example, say a company A has just bought another company B and there´s a full database integration planned for next year but in the meantime you need visibility over the whole company.

Both companies have Employees table(s) those tables are similar but because they come from different ERP systems they don't share all the fields. Both have identifiers for Employee and some other attributes like geographical information but also have some fields that do not match, like SSN and Bank Account.

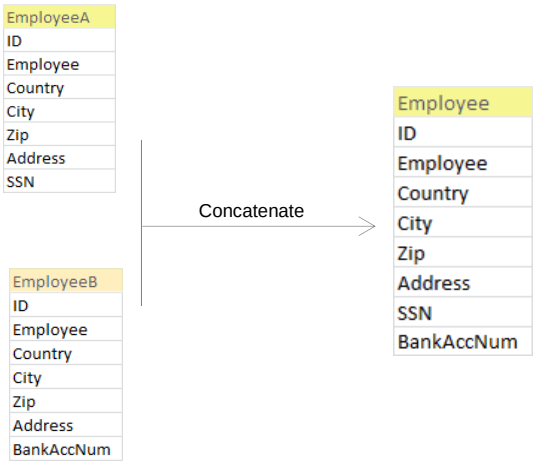
If you load those two tables straight to QlikView, it will create a Synthetic Key with the combination of common fields

Remember: Synthetic keys are neither bad nor good; they just are QlikView's attempt to resolve a data model linkage.

Alternatively to the straight load, you could force concatenation by including Concatenate statement before LOAD. QlikView's Concatenate function will allow you to perform the union even if the two tables are not equal and even if two tables are in separate databases.

```
LOAD
 ID,
 Employee,
 Country,
 City,
 Zip,
 Address,
 SSN
From Database A;

Concatenate LOAD
 ID,
 Employee,
 Country,
 City,
 Zip,
 Address,
 BankAccNum
From Database B;
```



This way you will be able to have a unified table of the both tables row by row, so if Employees A had 1000 records, and Employees B had 500 the concatenation of both will be one Employees table with 1500 records like this one.

ID	Employee	Country	City	Zip	Address	SSN	BankAccNum
1	Edward	US	Radnor	19111	Radnor St.	201201	-
2	Jane	US	Raleigh	18009	Raleigh St.	201205	-
A	David	SP	Madrid	28001	Madrid St.	-	91200004
B	Ricardo	PT	Lisbon	12122	Lisbon St.	-	322499999

This technique could be useful to Concatenate also **two fact tables** to simplify and rationalize the data model.

Let think we have Sales Orders and Purchase Orders, both tables are keep separate for performance optimization for standard SQL, but in QlikView we may want to have both tables as one fact table

Purchase Orders
VendorID
EmployeeID
OrderDate
OrderID
Discount
ProductID
Quantity
UnitPrice

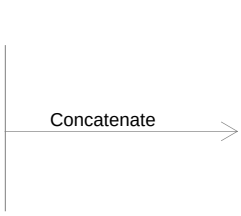
Sales Orders
CustomerID
EmployeeID
OrderDate
OrderID
Discount
ProductID
Quantity
UnitPrice

Purchase Orders and Sales Orders are almost identical except VendorID and CustomerID.

One possible linkage could be to force a concatenation by creating an alias for the non-matching fields. It will create a unified table of Orders including both, Sales and Purchases so to identify the origin I will include a flag (Type) into my table to identify the origin of the fields.

```
LOAD
 VendorID as ID,
 EmployeeID,
 OrderDate,
 OrderID,
 Discount,
 ProductID,
 Quatity,
 UnitPrice,
 'Purchase' as Type
From PurchaseOrders;

LOAD
 CustomerID as ID,
 EmployeeID,
 OrderDate,
 OrderID,
 Discount,
 ProductID,
 Quatity,
 UnitPrice,
 'Sale' as Type
From SalesOrders;
```



Orders
ID
EmployeeID
OrderDate
OrderID
Discount
ProductID
Quantity
UnitPrice
Type

Remember: This example will require Set Analysis in the expressions.

The concatenation of two or more fact tables will let you simplify your model and will let you create a star schema model like for an optimum performance.

QlikView

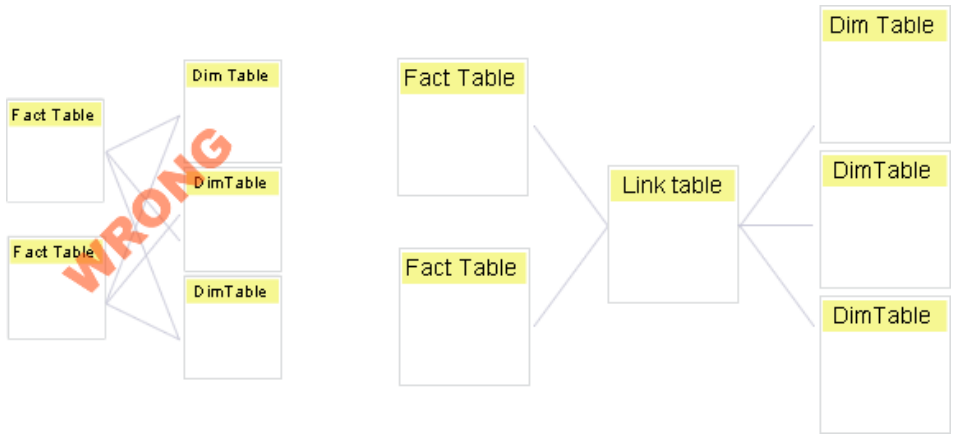
Link Table

> *"(…) a **junction table** is a database table that contains common fields from two or more other database tables within the same database. It is on the many side of a one-to-many relationship with each of the other tables. Junction tables are known under many names, among them **cross-reference table**, **bridge table**, **join table**, **map table**, **intersection table**, **linking table**, **many-to-many resolver**, **Link Table**, **pairing table**, **pivot table**, **transition table**, or **association table**. (…)."*
>
> *Source: Wikipedia*

This definition is mostly about relational SQL databases so if we adapt it for QlikView it could be something like the following.

> *Link Table: It´s a table that contains common fields from two or more tables (within the same database or not).*

The most common scenario for using Link Tables will be to replace synthetic keys and to avoid circular references (http://community.qlikview.com/docs/DOC-3451) by joining two or more fact tables against a common set of dimensions.

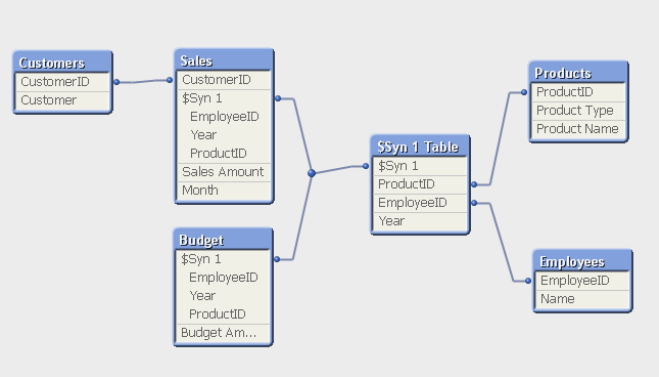


Let assume we are a Bicycle manufacturer who wants to visualize product sales versus budget over the years.

The tables we want to query in order to build our analysis are Budget, Sales, Employees, Products and Customers.

Note that sales are saved broken down by year/month but budget is defined on a yearly basis.

If we load these tables to QlikView it will solve the data model linkage with a synthetic table, $Syn1.



In the event of multiple fact tables QlikView allows us to create a central Link Table that only contains the existing data combinations.

Best Practice: Use AutoNumberHash128 for creating a compact memory representation of a complex key.

1st Load fact tables concatenating common fields in a field e.g. Key

```
Sales:
LOAD
    Year & '|' & EmployeeID& '|' & ProductID as Key,
    [Sales Amount],
    // EmployeeID,
    Month,
    //Year,
    //ProductID,
    CustomerID
FROM
Sales;

Budget:
LOAD
        Year & '|' & EmployeeID& '|' & ProductID as Key,
        [Budget Amount]
     //EmployeeID,
     //Year,
     //ProductID
FROM
Budget;
```

2nd Create the Link Table by loading the **distinct** values of the fact tables

```
LinkTable:
LOAD Distinct
        Year & '|' & EmployeeID& '|' & ProductID as Key,
        EmployeeID,
    Year,
    ProductID
FROM
Sales;
```

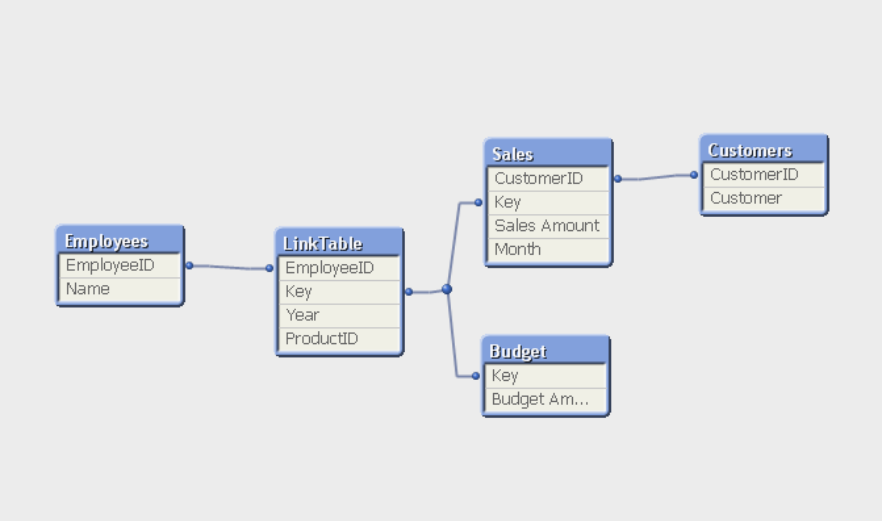
```
LinkTable:
LOAD Distinct
       Year & '|' & EmployeeID& '|' & ProductID as Key,
    EmployeeID,
    Year,
    ProductID
FROM
Budget;
```

3rd Load dimension tables

```
Customers:
LOAD Customer,
     CustomerID
FROM
Customers;

Employees:
LOAD Name,
     EmployeeID
FROM
Employees;
```

By following these steps we will create a schema with a central Link Table.



This will let us to avoid synthetic tables by creating a star schema model and as described before.

It's a good solution for those models with more than one fact table at different levels of detail (Sales and Budget in the example).

QlikView

Adding more parameters into the equation

Let recap for a while, we reviewed the use case scenarios for two methods of data modeling, Concatenate and Link Table.

While working with small sets of data any of the two approaches could be valid* but when we get into large volume data models you will need to consider also the size of fact tables as one parameter into the equation that will lead you to choose Link Table or Concatenate.

*Depending on what kind of analysis we want to create out of this data model.

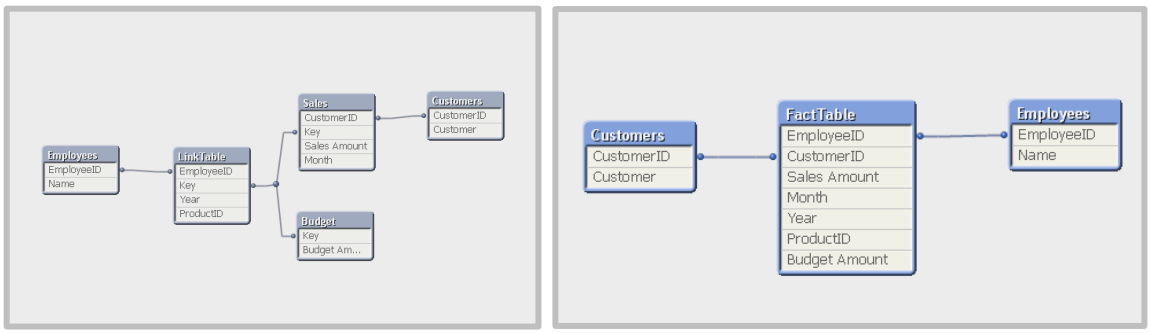
Let's consider the previous example:

As we discussed earlier one of the possible solutions could be to create a central Link Table with the common fields, but let assume that our Sales table contains dozens of millions of rows and Budget too, a central Link Table will still be the best solution?

By creating a Link Table you will be adding a new element to your data model and within the described scenario about multimillion rows this central table is going to be a new huge table to keep in memory.

Alternatively we could try concatenation instead but, this is going to be more efficient for this amount of data? And what if we have different granularity in the fact tables?

The answer to the performance question is yes, it's generally more efficient to Concatenate fact tables. By using Concatenate, we are avoiding a new table in the model with millions of records and on top of that we are reducing the number of query jumps. From five to just three in our example.



We still need to answer the second question, what if we have different granularity in the fact tables we want to Concatenate. Can we still use this method?

The answer again is yes concatenate may help you.

Let's review the model in the example, we had budget per year and employee and we are storing sales broken down by month. If we Concatenate the tables we will be able to compare sales and budget for the common fields, like, year, product or employee avoiding some of the common double counting issues that appear on relational SQL models.

Then, when I should use a Link Table?

> *"My example comes from pharmaceutical industry where the situation often is that you want to compare sales numbers from pharmacies with the sales calls that your sales representatives have made to physicians. (…)*
>
> *The sales numbers are per product package, i.e. a sub-group in the product dimension. Geographically the sales numbers are reported per physician, per pharmacy or per territory (depending on country), which only indirectly is connected to the sales representative.*
>
> *The sales calls are found in two tables: Visits and VisitDetails. These describe how a sales representative visits a physician and shows products. Several products can be shown in a sales call.*
>
> *Finally, the sales representatives have targets on how many calls they should make and how many products they should show. These are stored in the Targets table.*
>
> *An additional complication is the forked territorial dimension: Both physicians and sales representatives belong to territories, but there is no direct connection between a specific sales representative and a specific physician.*
>
> *The details of the challenge differ from country to country and from company to company, but the example still describes the general problem well.*
>
> *Loading these tables into QlikView as they are will obviously not work. There are far too many links that create circular dependencies. Instead the data must be remodeled into a snowflake scheme.*
>
> *For this, generic keys in combination with a master link table can be a great help."*
>
> *HIC. Generic Keys*

As we saw in this document, Link Tables and Concatenate are very useful solutions to handle your data.

Both methods have some advantages; with Link Tables you can keep a more understandable data model and in some cases it will let you create a set analysis free app.

Remember: In complex data models **hybrid approaches** are commonly used.

On the other hand Concatenate is a more simplistic approach with excellent performance; the bigger the data volume is the more important is to keep this in mind.