Goals and Agenda

- Introduce, define, and demonstrate Multiple Fact Table (a.k.a. Link Tables) Structure.
- Define limitations and benefits of Multiple Fact Table Design.
- Demonstrate use of generic keys.
- Demonstrate absolute market share using Set Analysis.
- Provide practice example for students to learn design techniques.

Qonnections

Global Partner Summit 2009

BARCELONA

The Typical Question

Posted: Oct 10 2006 - 03:17 AM
Subject: Help on Table definitions (and not only..)

Hi,
i'm working with QlikView Enterprise 7.5 and with sql server tables (loaded via ODBC) that contains my intranet data (calls, planning, fax and so on..)

I have these tables:

Users (the intranet users)
Customers (our customers)
Calls (the daily calls)
Planning (our daily planning)

i'm trying to build a working sheet that allow me, selecting data, users and customer to show the calls and appointments of that users with that customer. Well i'm in trouble.. i don't understand how to join correctly the Calls and the Planning table to users and customers.

these are my tables:

Users:
Id
UserId
Password
Level
LastLogin
Active
Group
CompleteName
IsAGroup

Customers:
CUSTOMER_ID
CUSTOMER_NAME
ADDREESS
CAP
CITY
PR
P_IVA
COD_FISCALE
PHONE
FAX

Calls:
Date
CUSTOMER_ID
Operator
UserId (ID of Users table)
Argument

Planning:
UserId (ID of Users table)
Date
Time
CUSTOMER_ID
Note

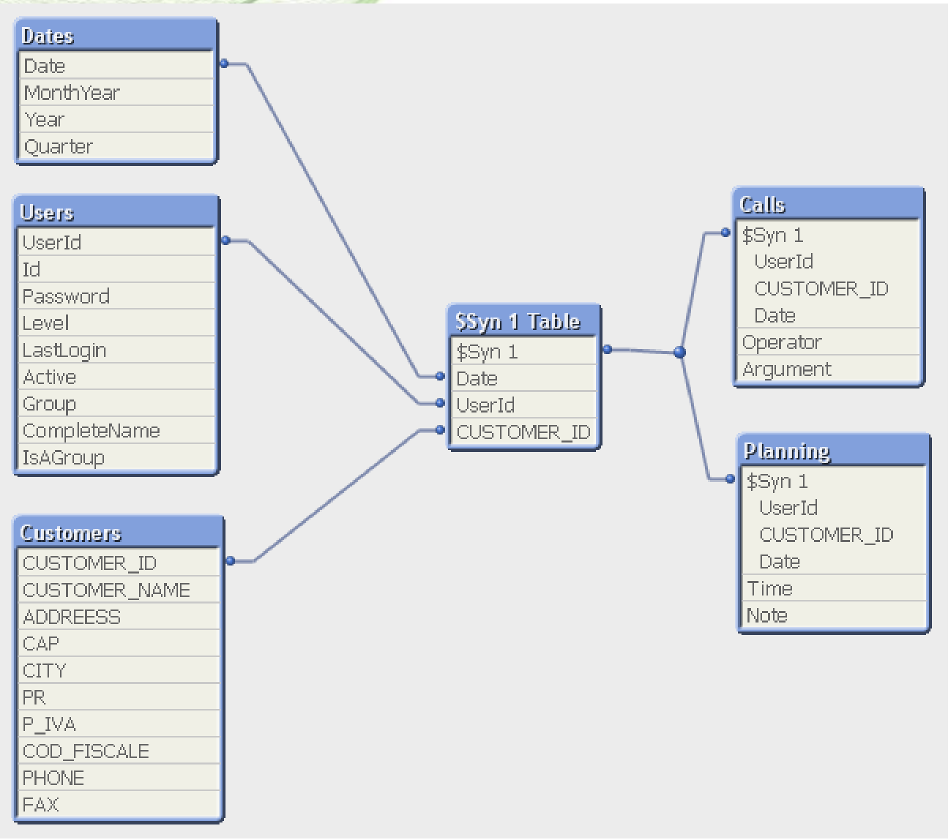
Qonnections

Global Partner Summit 2009

BARCELONA

The Typical Question – A Possible Solution

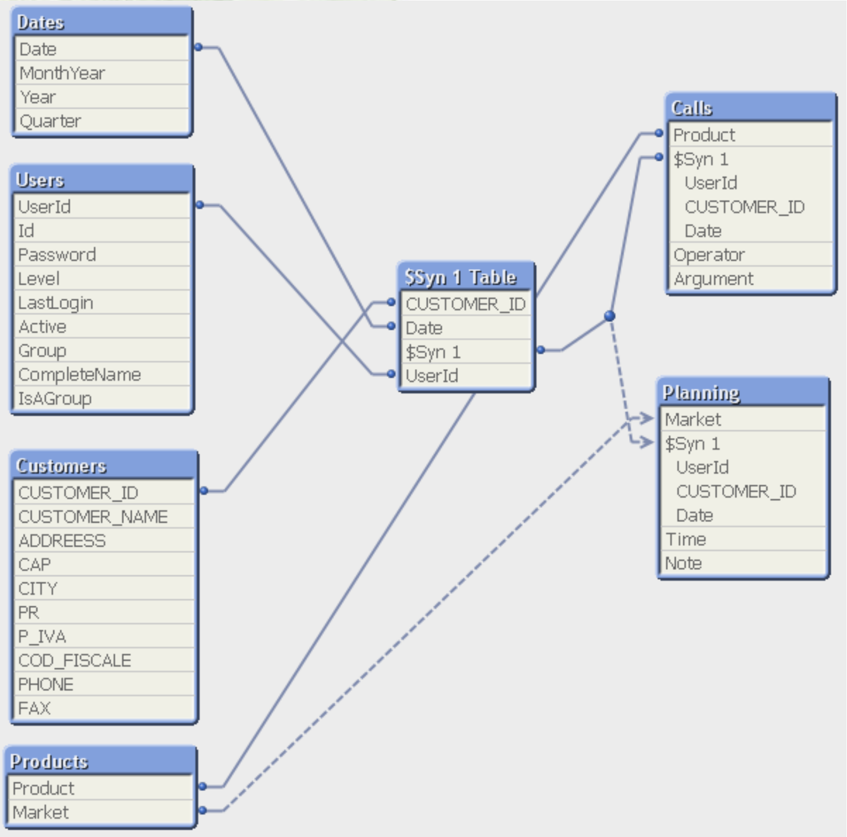
This structure will load, as is, into QlikView cleanly, with no circular loops, and 1 synthetic key:



The Typical Question – Add Product/Market

However, if Product information is added, and Product is added to Calls, Market added to Planning:

The structure quickly degrades to a circular loop and a loosely coupled table



Dates
- Date
- MonthYear
- Year
- Quarter

Users
- UserId
- Id
- Password
- Level
- LastLogin
- Active
- Group
- CompleteName
- IsAGroup

Customers
- CUSTOMER_ID
- CUSTOMER_NAME
- ADDREESS
- CAP
- CITY
- PR
- P_IVA
- COD_FISCALE
- PHONE
- FAX

Products
- Product
- Market

$Syn 1 Table
- CUSTOMER_ID
- Date
- $Syn 1
- UserId

Calls
- Product
- $Syn 1
  - UserId
  - CUSTOMER_ID
  - Date
- Operator
- Argument

Planning
- Market
- $Syn 1
  - UserId
  - CUSTOMER_ID
  - Date
- Time
- Note

Qonnections

Global Partner Summit 2009

BARCELONA

Problems with traditional Data Structures

- Synthetic keys
- Greater possibility of circular loops
- Selection of one type of data precludes other types of data
- Difficult to add new types of data, especially fact
- Does not support Generic Keys cleanly
- Relationships are typically complex and hard to understand

Multiple Fact Table Structure

An alternative approach for the QlikView logical data structure in complex applications.

Multiple Fact Table Structures are designed to avoid the problems with traditional structures by creating a star schema in QlikView, separating data into *Dimensions* and a *single Fact table*. This is cleaner, easier, and more efficient than multiple Fact tables in a traditional link table structure.

Then, associating these two types of data with “linking” tables to obtain Generic Key benefits

Multiple Fact Data Structures Design Technique

Qonnections

Global Partner Summit 2009

BARCELONA

Design Technique - Dimensions

- Dimension tables contain relatively static component data, such as Products, Customers, Employees, and Dates.
- All selectable fields (including chart dimensions) should reside in these tables (and optionally in a secondary link table).
- These tables will almost always have a single key field. It is possible that some dimension tables will share the same key field.
- Dimension tables can also include "counting“ tables, which contain the DISTINCT key field values and a constant (typically 1) used to sum the current number of selected values, e.g. sum(Physician_Count)

Note: It is generally good practice in QlikView to try to stay as close to the STAR schema as possible, and avoid normalizing (or snowflaking) dimension tables. This will result in the shortest possible chain lengths (traverses) when resolving a chart expression.

Design Technique – Dimensions Example

Example:

```
D_Territory:
Load
        Territory as [%Territory],
        Territory,
        District,
        Region
FROM [Sales Geography.csv] (ansi, txt, delimiter is ',', embedded labels);
```

Exercise 1

1. Open LinkTable85_Exercise.qvw
2. Review Dimension tab in load script

Qonnections

Global Partner Summit 2009

BARCELONA

Design Technique - Fact

- The Fact table contains transaction data. The data is almost always time-based, and each row is typically defined by combinations of dimension keys, e.g. Date, Product, Customer (Physician in this example).
- The final load of the Fact Table data will contain all key fields. Use generic for key fields that do not exist in a particular fact table set of records.
- Other fields will include measure fields, e.g. TRx (sales amount) that will be aggregated in charts and other UI objects. In addition, flag fields should be included here.
- The Fact Table should NEVER contain selectable fields, since selection in one set of fact table records will cause almost all other fact table records to become non referenced (due to specific keys).

Design Technique – Fact Example

Example:

```
All_Facts:
Load
// KEYS
     DateIndex as [%Date],
     $(v_A) $(v_S) PhysicianID as [%Territory_Physician],
     ProductID as [%Product],
     $(v_A) as [%DetailPositionLink],
// FLAGS
     -1*(right(ProductID,2)='01') as flag_TRx_Company,
     -1*(DateIndex < 4) as flag_curr_3m,
     -1*(DateIndex > 3 and DateIndex < 7) as flag_prev_3m,
     applymap('M_Current_Year',DateIndex,0) as flag_curr_y,
// MEASURES
     TRx
FROM Sales_Source.qvd (qvd)
WHERE
     TRx <> 0;
```

Generic Key for sales territory

Define flags here to avoid double counting w/ generic keys

Should rarely, if ever need zero values

Exercise 2

1. Open LinkTable85_Exercise.qvw
2. Add additional Fact table records from Calls_Sample.qvd
   - Use [Fact Table: Call Activity] sheet
   - Concatenate to All_Facts table
   - Define all standard key fields
   - Define all standard flag fields
   - Use 1 as Sample_Count
   - Include SampleQuantity as a measure

Design Technique – Secondary Link Tables

- Secondary Link tables are optional, but serve a number of purposes.
- These tables define ***relationships*** between dimension and fact tables as well as between multiple dimension tables. For example, the relationship between territories and physicians will be defined in the secondary link table L2_Alignment. This feature allows a generic key that still honors the inherent data relationships and hierarchies.
- This is where all generic keys are defined. A generic key is a specific value (typically 'ALL', set in variable [v_A]) that replaces the existing value in the portion of the field (or the field itself) that connects to the FACT table. The generic value should never exist in the key field link to the dimension table.
- Selection fields are allowed in these tables, but are generally rare.
- Recommendation is to use DISTINCT on all loads

Design Technique – Secondary Link Tables Example

Example:

```
L2_Detail_Position:
Load DISTINCT
          [%DetailPosition] as [%DetailPositionLink],
          [%DetailPosition]
Resident
          D_Detail_Position;

// GENERIC DETAIL POSITION

CONCATENATE (L2_Detail_Position)
Load DISTINCT
          $(v_A) as [%DetailPositionLink],
          [%DetailPosition]
Resident
          D_Detail_Position;
```

This is link to Fact table. It should match Fact table keys.

This is the Generic value.

This key links directly to the Dimension table. These types of keys should never be generic.

Generic Keys

Qonnections

Global Partner Summit 2009

BARCELONA

Generic keys

- The main purpose of generic keys is to allow the use of a standard set of Fact table keys without requiring key values for all dimensions and all types of fact data.
- A generic key is a specific value (typically 'ALL', set in variable [v_A]) that replaces the existing value in the portion of the field (or the field itself) that connects to the Fact table.
- Make sure that the value you use will never occur in your actual data.
- The generic value should never exist in the key field link to the dimension table.

Generic keys - Single Key Field

To create a generic value for a single key, simply repeat each record with the generic value.

%D_Dim	%D_Fact
1	1
2	2
3	3
4	4
5	5
6	6

1	ALL
2	ALL
3	ALL
4	ALL
5	ALL
6	ALL

Generic keys - Multiple Key Fields

To create a generic value for multiple keys, reload data for all required combinations of generic data.

%T	%C	%TC_Fact
X	1	X1
X	2	X2
Y	3	Y3
Y	4	Y4

X	1	X_ALL
X	2	X_ALL
Y	3	Y_ALL
Y	4	Y_ALL

X	1	ALL_1
X	2	ALL_2
Y	3	ALL_3
Y	4	ALL_4

X	1	ALL_ALL
X	2	ALL_ALL
Y	3	ALL_ALL
Y	4	ALL_ALL

Exercise 3

1. Open LinkTable85_Exercise.qvw
2. Add load for generic Territory in table [L2_Alignment]
   - Use [Secondary Link Tables] sheet
   - Concatenate to L2_Alignment table
   - Use defined variables for generic key and separator

Calculating Absolute Market Share Using Set Analysis

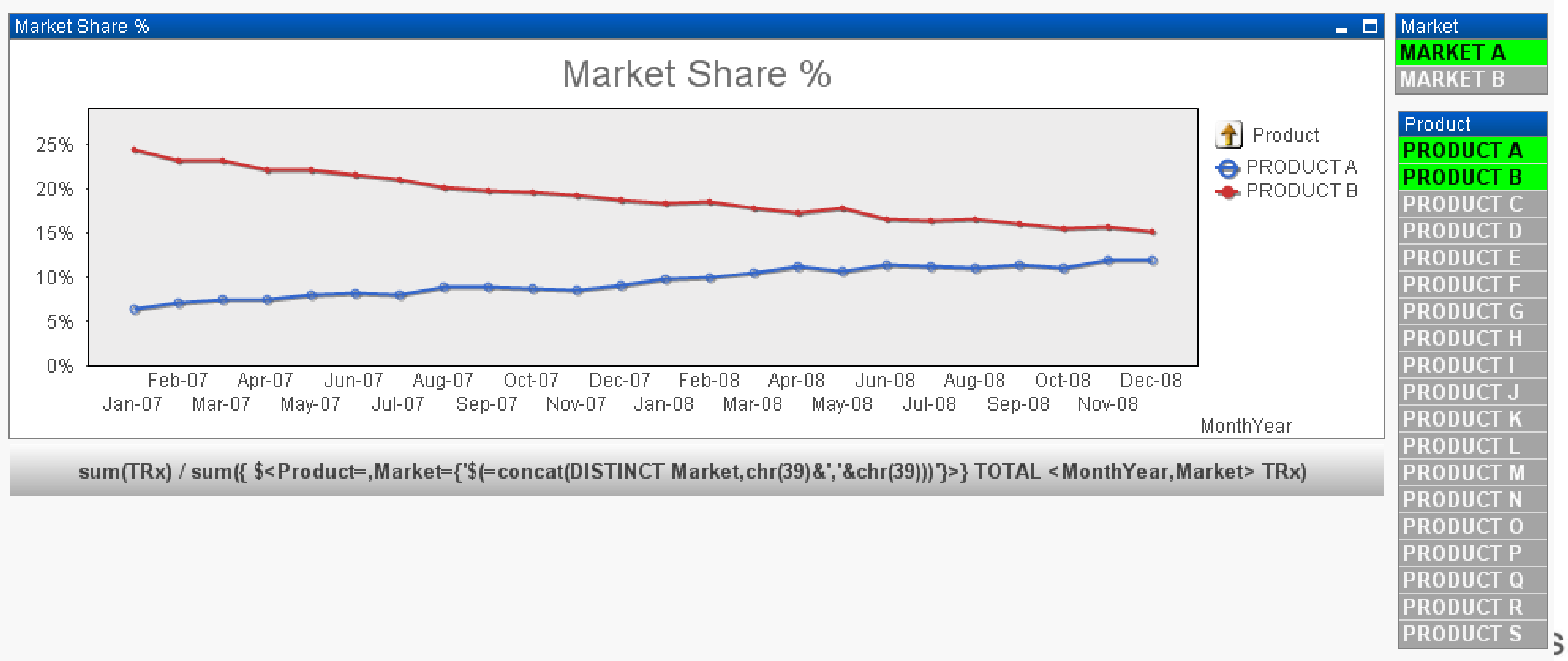
Qonnections

Global Partner Summit 2009

BARCELONA

Set Analysis

With Set Analysis, we no longer need to pre-aggregate denominator values for market share. Use a Set that defines the current selection minus any product selection for the denominator. In this case, we should also add a select for the associated Market in order to get the correct result regardless of selection state.



```
sum(TRx) / sum({ $<Product=,Market={'$(=concat(DISTINCT Market,chr(39)&','&chr(39)))'}>} TOTAL <MonthYear,Market> TRx)
```

Market
- MARKET A
- MARKET B

Product
- PRODUCT A
- PRODUCT B
- PRODUCT C
- PRODUCT D
- PRODUCT E
- PRODUCT F
- PRODUCT G
- PRODUCT H
- PRODUCT I
- PRODUCT J
- PRODUCT K
- PRODUCT L
- PRODUCT M
- PRODUCT N
- PRODUCT O
- PRODUCT P
- PRODUCT Q
- PRODUCT R
- PRODUCT S

Set Analysis

Calculating Market Share:

Market Share = Sum of Product Sales / Sum of all Products in Market Sales

In our example:

Sum of Product Sales = Sum (TRx)

Sum of all Products in Market Sales = Sum ({ $<Product=>} TRx)

Since the calculation is taking place within a chart, we need to account for the dimensions using TOTAL <MonthYear,Market>

{ $<Product=,Market={***'$(=concat(DISTINCT Market,chr(39)&'***,'&chr(39)))'}>}

Since it is possible that no Market is selected, we need to account for this in the Set Analysis by “selecting” the possible markets. Use the Concat() function to list the Market values associated with each Product (value in the dimension)

Set Analysis

Set Analysis Observations:

- Be careful of syntax
- Avoid using key fields
- Set Analysis seems to use chart cache a little different
- Understand use of **TOTAL** when working within chart dimensions
- Be careful of using **Suppress When Value Is Null** on chart dimensions

Exercise 4

1. Open LinkTable85_Exercise.qvw
2. Client would like to add Goal data to this document. This is a rolling comparison of 24 months of sales data used to compare to the 24 months of current data for Company products.
3. Goal data is located in Sales_Goal.qvd
4. Goal data has Product and Physician (Customer), but no time dimension.
5. Add a new tab in load script, and add Goal data to All_Facts
6. Add script code to account for generic dates
7. Add a bar chart comparing sum(TRx) to sum(TRx_Goal) by Product_Company