PreSales Academy

Data Modelling

QlikView

Objectives

- Defining Data Models
- Understand how QlikView is Different from SQL
- Understand Data Warehousing Theory
- Adopt Applicable QlikView Data Modeling Best Practices

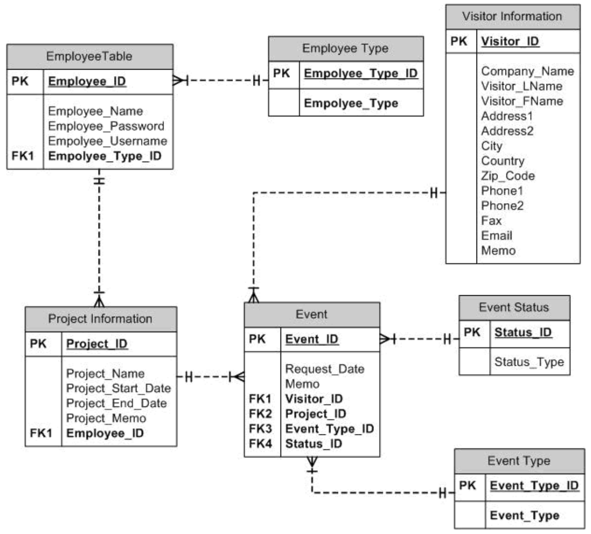
What do we mean by Data Model?

Traditional definition:

- A traditional data model is a **visual representation** of the people, places and **things of interest to a business** and is composed of **symbols that represent the concepts and their business rules**.

- Like a building architect, who creates a series of diagrams or blueprints from which a house can be constructed, a data modeler/architect creates **diagrams from which a database may be built**.

- This will ***NOT*** be the topic of our discussion today.

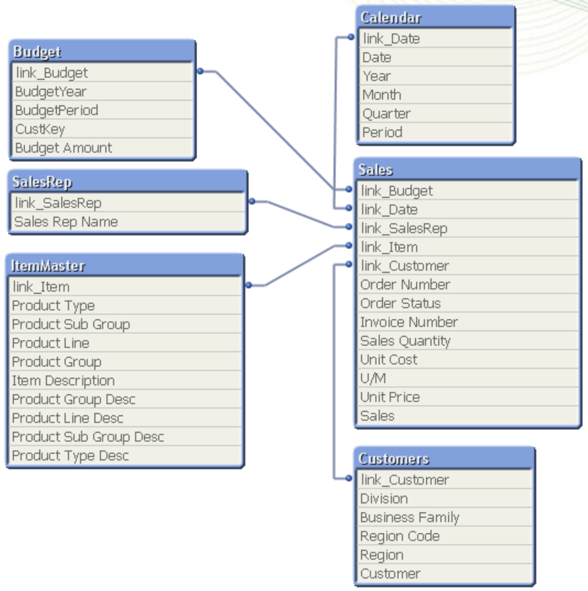


What do we mean by Data Model?

QlikView definition:

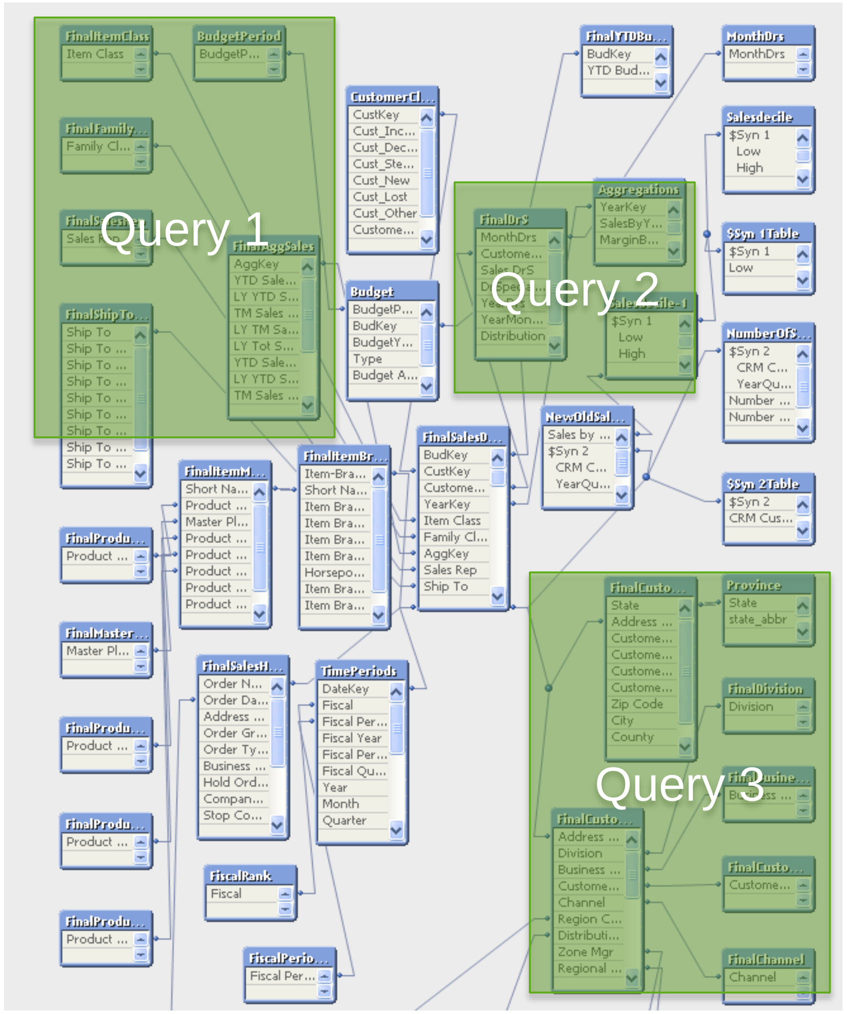
- **A QlikView data model is the representation of data you have loaded**.
- When you load your data in to the QlikView application, a data model will be created based on the tables and columns you have in your script and also the names of the columns and any resident loads and joins you have previously defined.
- You will of course be driven by the type and structure of your data sources.
- These sources and the underling data will have to **be manipulated within the script to deliver the Data Model that best suits your data for both performance and usability.**

- This will be our topic today.



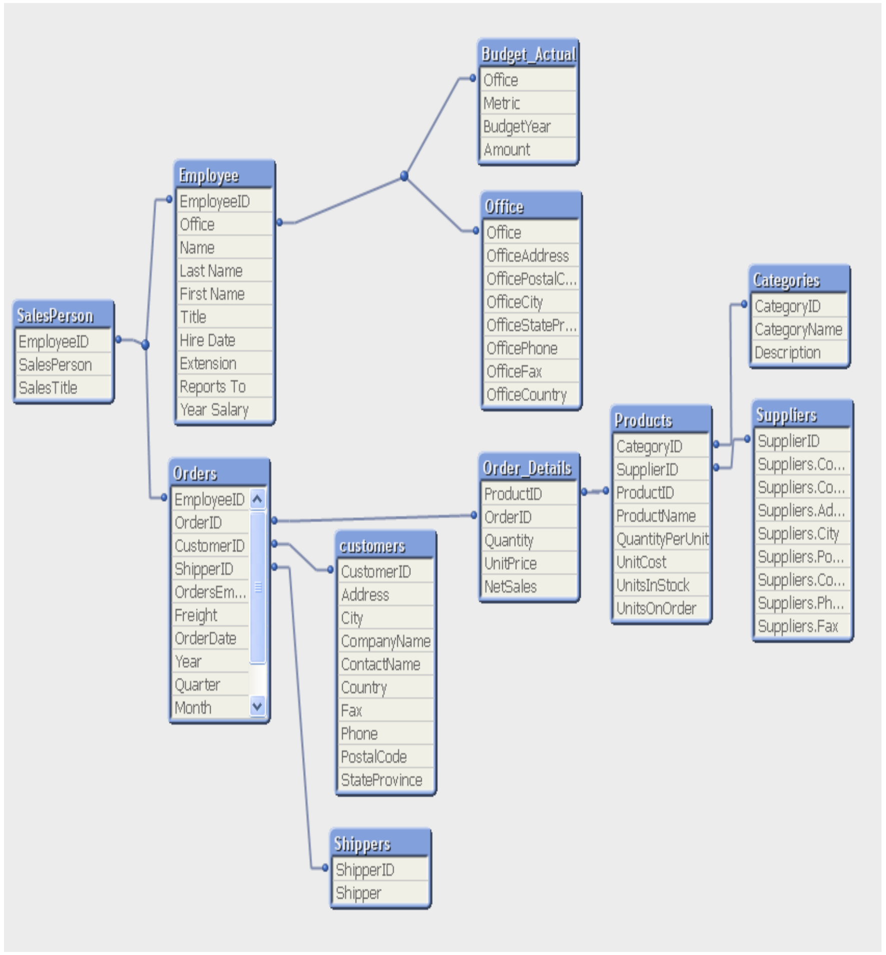
QlikView data models

QlikView is not SQL (SQL Schemas)



- SQL take a large schema and queries a subset of tables.
- Each query creates a temporary "Schema" of only a few tables.
- Query result sets are independent of each other.

QlikView is not SQL (QV Schemas)



- QlikView builds a smaller and more reporting friendly schema from the transactional database.
- This schema is persistent and reacts as a whole to user “queries”.
- A selection affects the entire schema.

QlikView is not SQL (Aggregation and Granularity)

Store Table

Store	FloorArea
A	1000
B	800

Sales Table

Store	Product	Price	Date
A	1	$1.25	1/1/2010
A	2	$0.75	1/2/2010
A	3	$2.50	1/3/2010
B	1	$1.25	1/4/2010
B	2	$0.75	1/5/2010

Select * From Store, Sales Where Store.Store = Sales.Store will return:

Floor Area	Store	Product	Price	Date
1000	A	1	$1.25	1/1/2010
1000	A	2	$0.75	1/2/2010
1000	A	3	$2.50	1/3/2010
800	B	1	$1.25	1/4/2010
800	B	2	$0.75	1/5/2010

Sum(FloorArea) will return: **4600**

If you want the accurate Sum of FloorArea in SQL you cannot join on the Sales table in the same Query!

QlikView is not SQL (Benefits)

- QlikView allows you to **see the results of a selection across the entire schema** not just a limited subset of tables.

- QlikView will **aggregate at the lowest level of granularity in the expression** not the lowest level of granularity in the schema (query) like SQL.

- This means that **QlikView will allow a user to interact with a broader range of data than will ever be possible in SQL!**

QlikView is not SQL (Challenges)

- Several SQL queries can join different tables together in completely different manners.
- In QlikView there is only ever One way tables join in any one QlikView file.
- This means **that Schema design is *much* more important in QlikView!**

Audience participation!

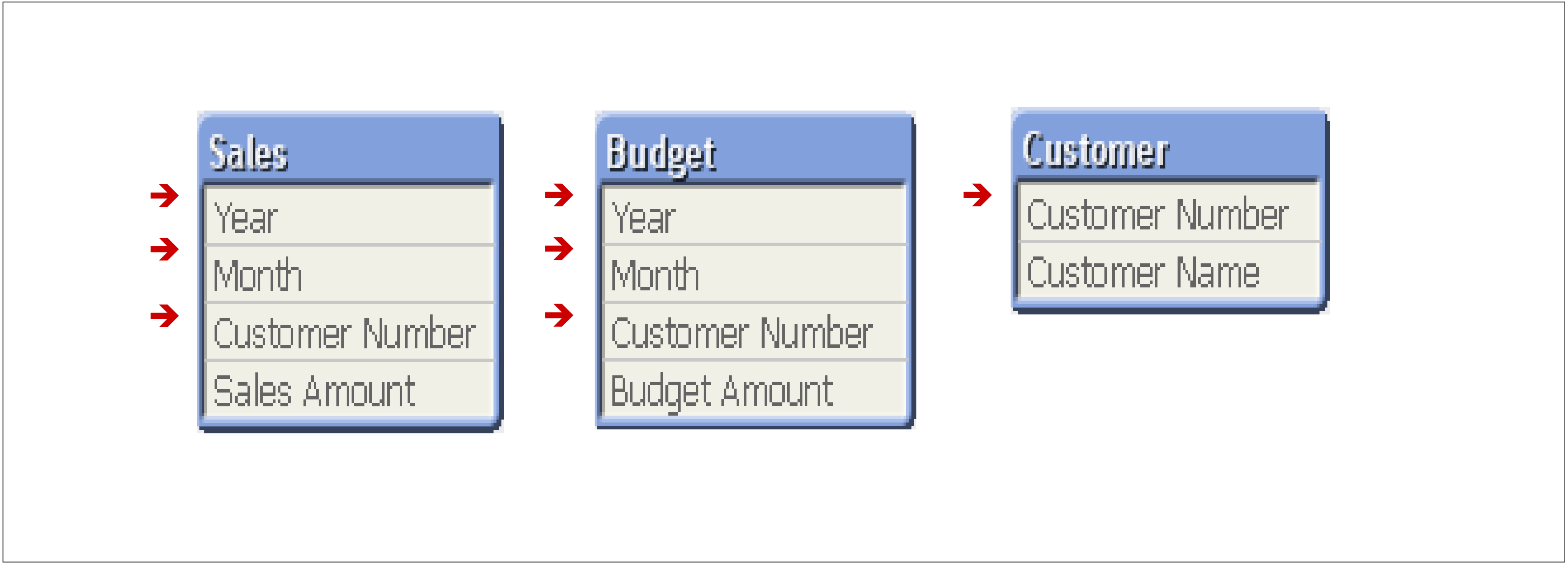
- What challenges have you encountered with basic data modeling in QlikView?

- Most common initial challenges :
  - Synthetic keys
  - Circular references

Synthetic Keys

Synthetic Keys

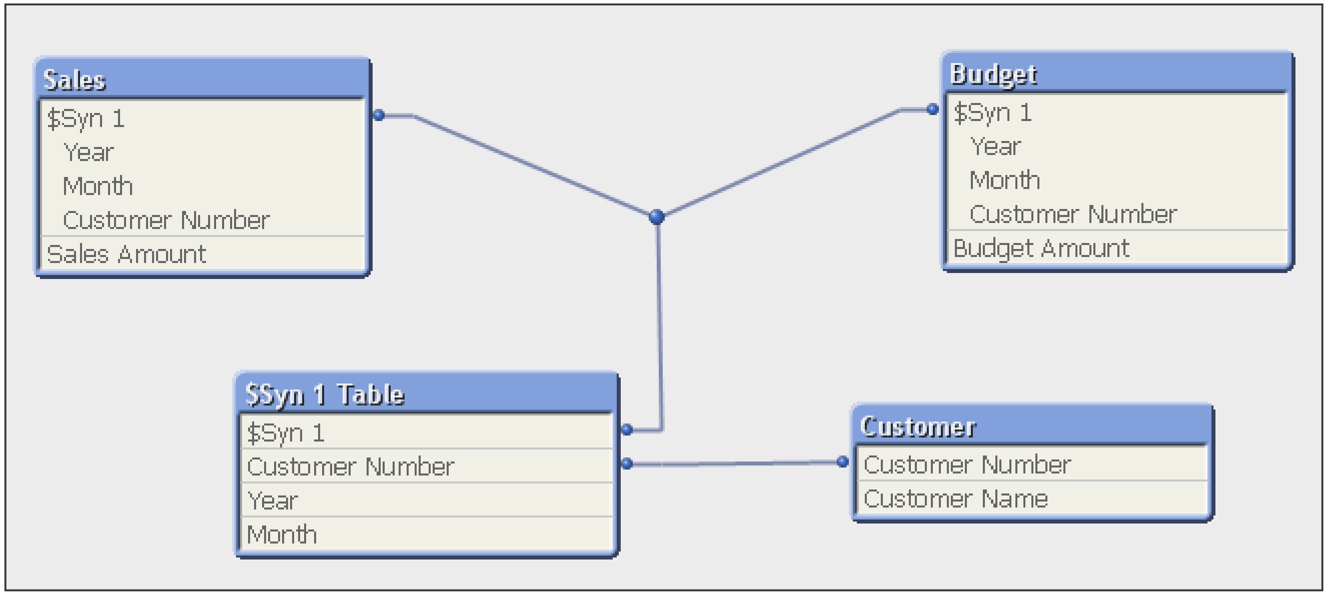
- When there is more than one field in common between tables



➔ If you load as is, then…

Synthetic Keys

➔ QlikView creates synthetic keys



Synthetic Keys: Challenge

Q: What is a synthetic key?

A: It is a field that contains all possible combinations of common fields among tables

Q: Is a synthetic key bad?

A: No, but try to avoid it. It is generated by QlikView. That means you could lose the control over it when you have many of them.

Challenge

- How many ways are there to resolve a synthetic key?

4

1. An ANSI JOIN
2. A Concatenated Key
3. Concatenated Tables
4. A Link table

Synthetic Keys Solutions - Join

Q: How do I avoid a synthetic key? - #1

A: Join tables by common fields

```
Sales:
Load
   Year,
   Month,
   [Customer Number],
   [Sales Amount]
FROM Sales;

LEFT JOIN Load ←
   Year,
   Month,
   [Customer Number],
   [Budget Amount]
FROM Budget;
```

```
Customer:
Load
   [Customer Number],
   [Customer Name]
FROM Customer;
```

Problem!

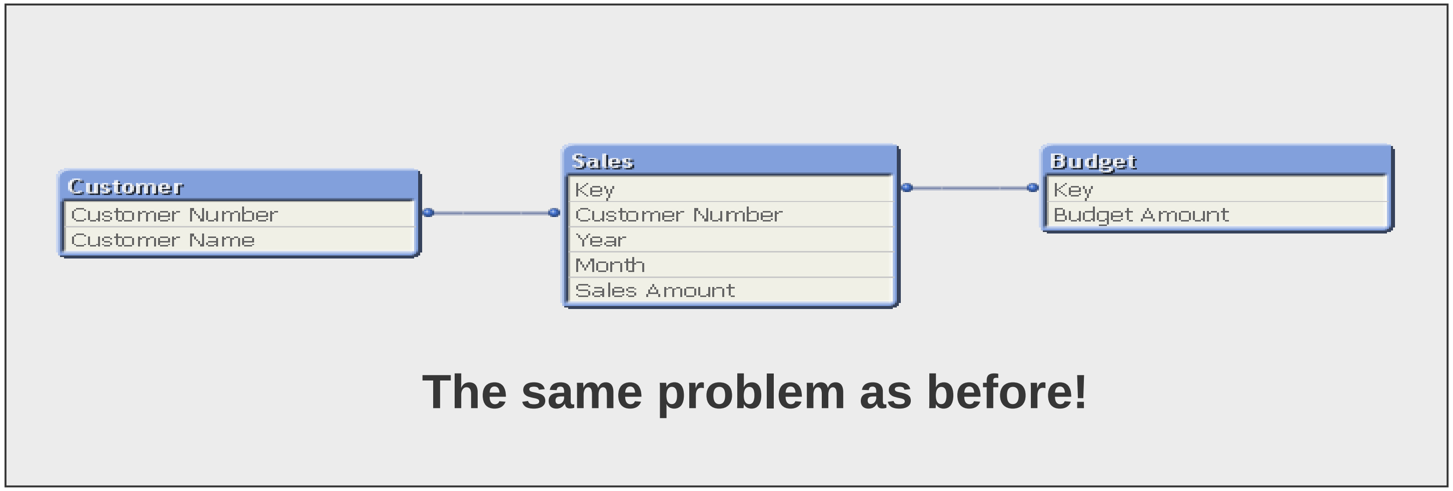
- Not getting all the data from Budget table results in missing months for the rest of the year
- Even if joining the sales table to budget table, still missing customers' activities who are not budgeted
- May become a problem if tables don't have a one-to-one relationship

Synthetic Keys Solutions – Create key

Q: How do I avoid a synthetic key? - #2

A: Create a key on your own by concatenating the common fields

```
Year & '_' & Month & '_' & [Customer Number] as Key
```



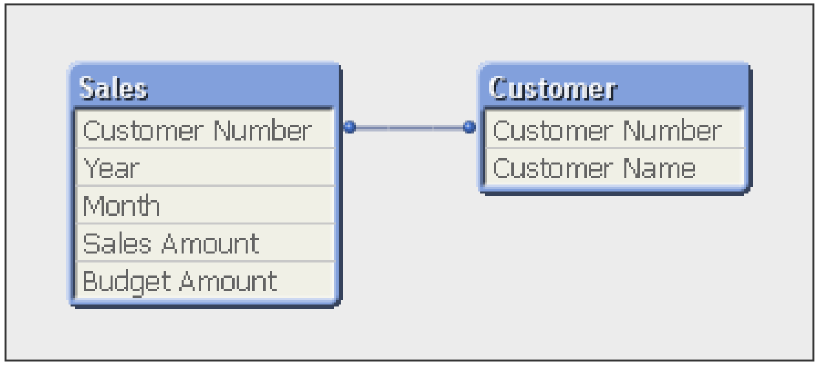
The same problem as before!

Synthetic Keys Solutions – Auto concatenate

Q: How do I avoid a synthetic key? - #3

A: Combine (concatenate) the tables so you have all the possible values

```
Sales:
Load
  Year,
  Month,
  [Customer Number],
  [Sales Amount],
➔ Null() as [Budget Amount]
FROM Sales;
Budget:
Load
  Year,
  Month,
  [Customer Number],
➔ Null() as [Sales Amount],
  [Budget Amount]
FROM Budget;
```



Note:

- **When QlikView finds multiple tables with the exact same fields, it combines them into one table automatically**
- **Create empty fields (dummy fields) using null() function for missing ones in each table**

Synthetic Keys

Q: What is the benefit of combining tables into one?

A: Guaranteed to keep all the data in a table.

Q: What is the benefit of using Auto-Concatenate?

A: When some fields are misspelled, or when some fields are left out by mistake, then they could be easily identified (synthetic keys will appear).

Q: Do we use the concatenation method often?

A: Yes. Its the single most widely utilised QlikView method for resolving synthetic keys.

Q: Is there a way to avoid automatic concatenation?

A: Yes. Use the syntax “Noconcatenate Load” instead of “Load”. Gives you more control.

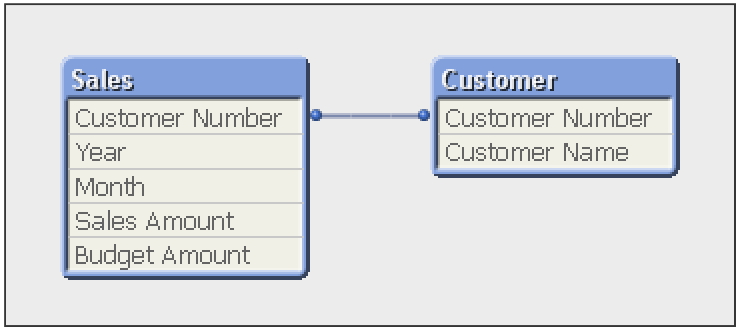
Synthetic Keys Solutions – Forced concatenate

Q: What is Forced Concatenate?

A: QlikView creates empty fields automatically so there is no need to create dummy fields manually

```
Sales:
Load
   Year,
   Month,
   [Customer Number],
   [Sales Amount]
FROM Sales;

Budget:
CONCATENATE Load ←
   Year,
   Month,
   [Customer Number],
   [Budget Amount]
FROM Budget;
```



Sales	Customer
Customer Number	Customer Number
Year	Customer Name
Month	
Sales Amount	
Budget Amount	

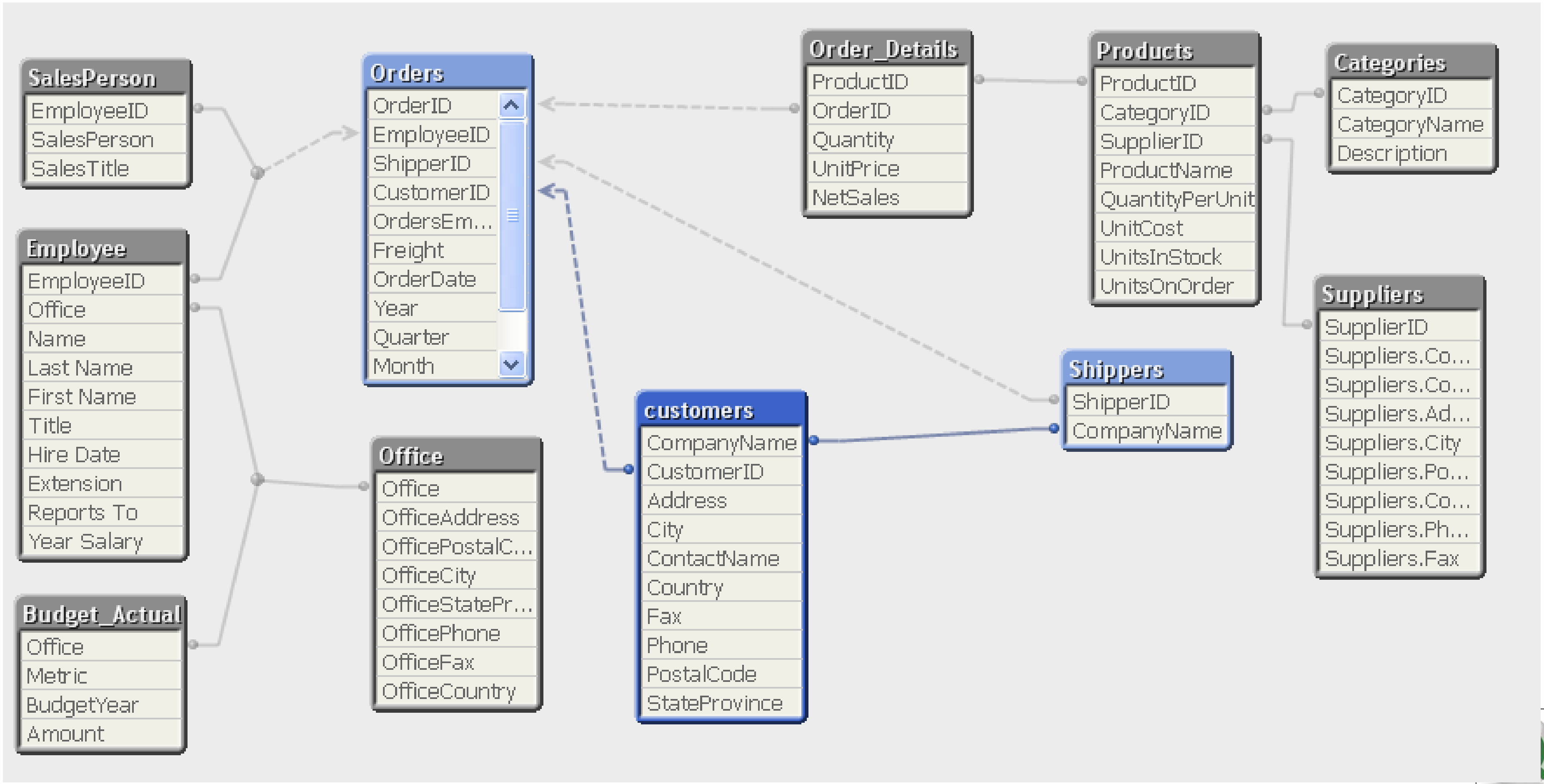
Note:

- This script will end up with two tables. It is the same structure as Auto-Concatenate method

Circular references

Circular References

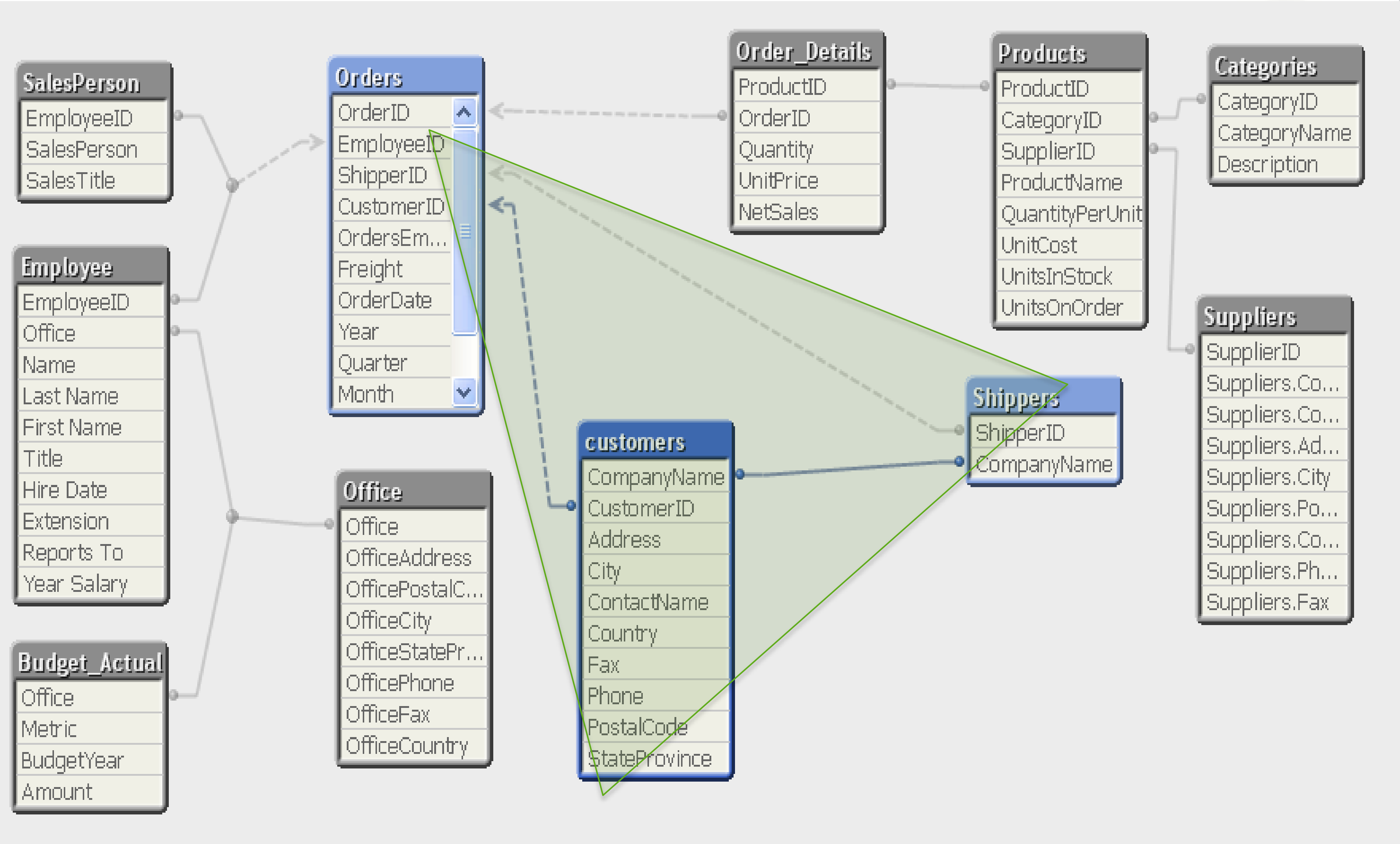
- Anytime an area is enclosed in the table viewer you will encounter a circular reference, for example if you have two fact tables which share a common dimension table.



Circular References

- **Circular References are common** in QlikView because you get only one set of join relationships per QlikView file.
- When you get a circular reference ask yourself if you could live without one instance of the field that is causing the extra association (such as a duplicated field). **If you can, rename it or remove it.**
- Otherwise you may have **to resort to concatenation or a link table to remove** the circular reference.
- Don’t kill yourself with technical link tables if you don’t have to!

Circular Reference Solutions – Challenge

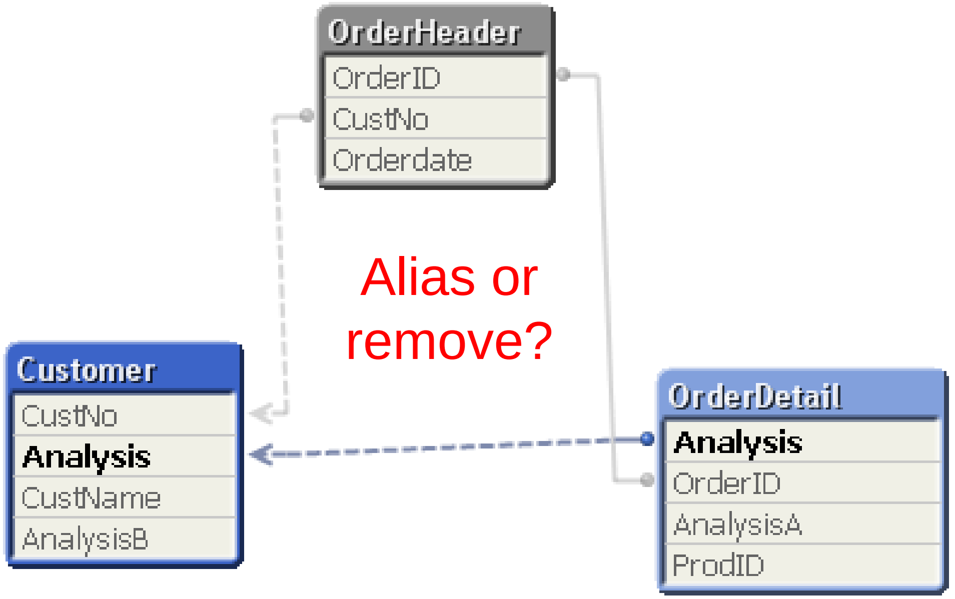


Circular Reference Solutions - Answer

- It depends on the business logic in most cases

- In our example the question to ask is even more basic:
  - Is the Shipper Company Name the same as the customer company name as this look the most likely candidate to modify in order to remove the circular reference

- The example below is equally common, and revolves around the business logic and reporting requirements, in this case relating to analysis fields.



Star schema

The Star Schema Approach

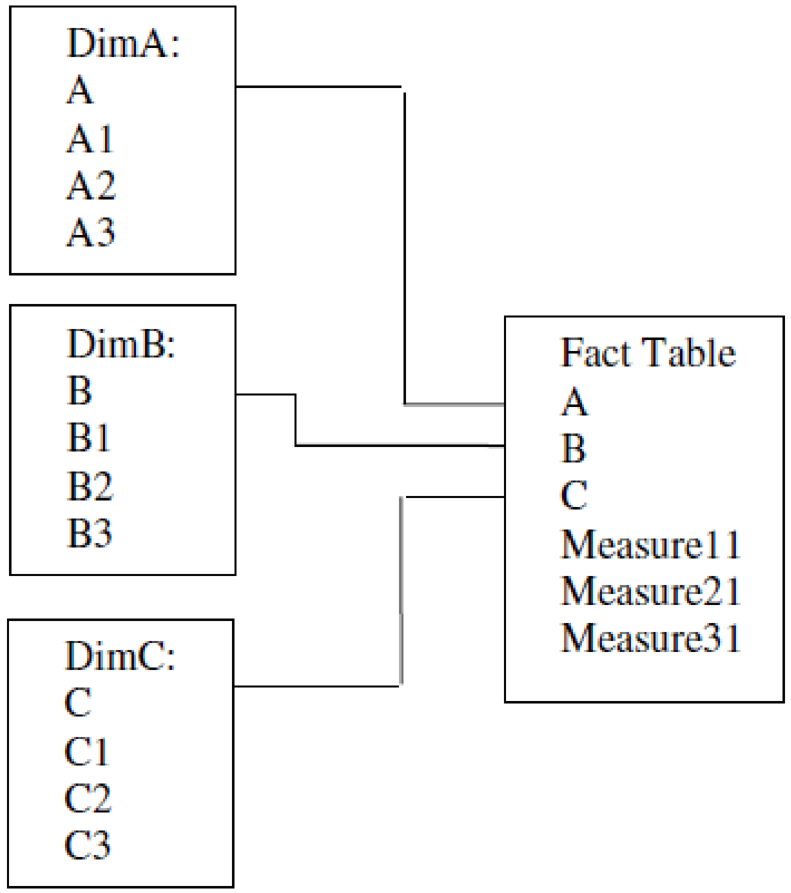
- The standard layout and structure of data presentation is the Star Schema. QlikView is generally most efficient when working in this space.

The star schema (sometimes referenced as star join schema) is the simplest style of data warehouse schema. The star schema consists of a few fact tables (possibly only one, justifying the name) referencing any number of dimension tables. The star schema is considered an important special case of the snowflake schema.

(Source, Wikipedia - http://en.wikipedia.org/wiki/Star_schema)

- Within a Star schema model, the event data (transactions) reside in a central “Fact Table” and the attributes of the event reside in separate “dimension tables”. The following diagram shows the basic layout…

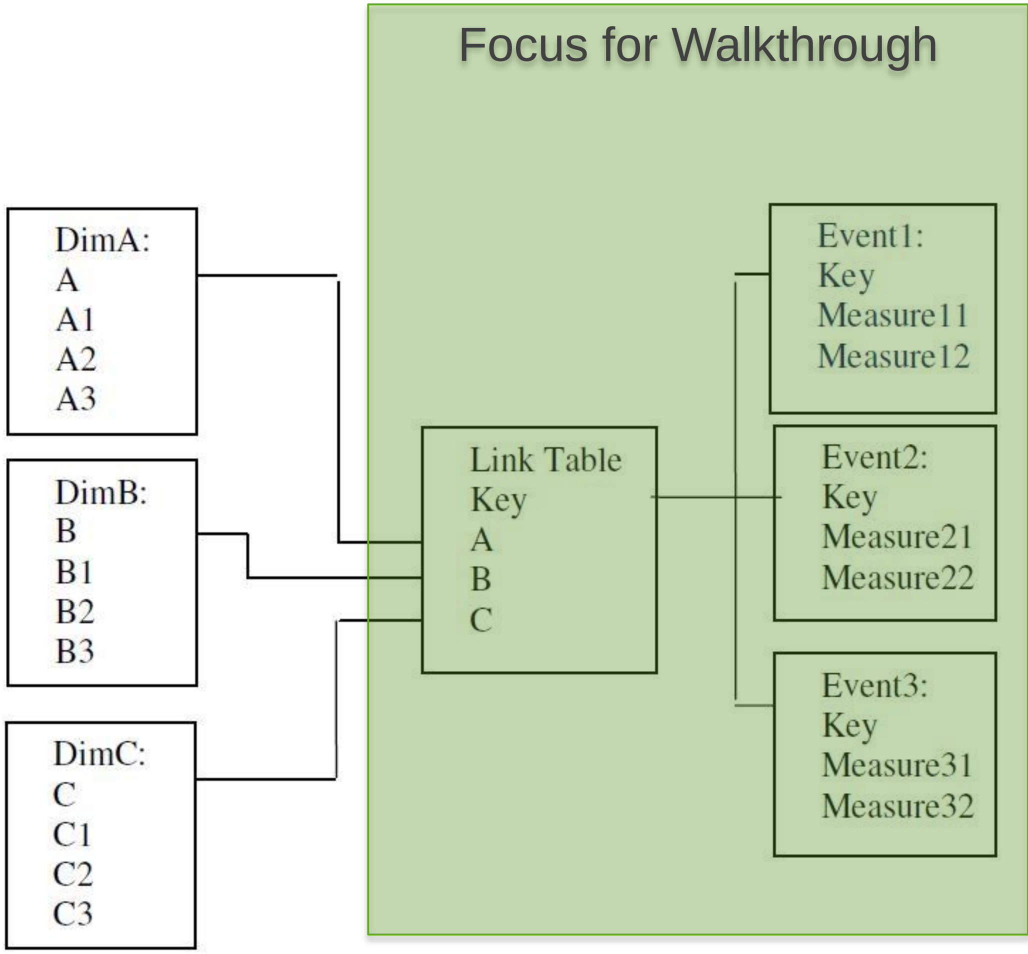
The Star Schema Approach



- This model works well in a **simplistic, single event scenario**. But as QlikView can handle multiple data sources from many different source systems and files, we have to work with multiple event scenarios, or many fact tables.

Central Link Table (Event Space)

- In the event of **multiple fact tables** QlikView allows us to create a **central link table** that only contains the existing data combinations.
- Instead of *Joining* the tables, the **event dimensions can be CONCATENATED** in to one central Link table.
- This link table can then be linked back to **the event measures one side and the dimension tables on the other.**


Focus for Walkthrough
DimA: A A1 A2 A3
DimB: B B1 B2 B3
DimC: C C1 C2 C3
Link Table Key A B C
Event1: Key Measure11 Measure12
Event2: Key Measure21 Measure22
Event3: Key Measure31 Measure32

When do I use a link table?

Q: When do I use a link table?

A: When there are common fields in multiple tables (a synthetic key exists) but most of the fields from each table are not shared

Example 1:

```
Sales:
Load
   Year,
   Month,
   [Customer Number],
   [Sales Amount]
FROM Sales;
```

```
Budget:
Load
   Year,
   Month,
   [Customer Number],
   [Budget Amount]
FROM Budget;
```

```
Customer:
Load
   [Customer Number],
   [Customer Name]
FROM Customer;
```

- In this example, a concatenation of FACT tables would be the preferable solution, although a basic link table solution is also valid.

When do I use a link table?

Example 2:

```
Sales:
Load
   Year,
   Month,
   Branch,
   [Item Number],
   [Customer Number],
   [Invoice Number],
   [Order Number],
   [Salesman Number],
   [Invoice Date],
   [Sales Amount],
   [Sales Qty],
   [Cost Amount],
   [Margin Amount],
   [Unit of Measure]
FROM Sales;
```

```
Inventory:
Load
   Branch,
   [Item Number],
   [On Hand Qty]
FROM Inventory;
```

```
Purchasing:
Load
   Year,
   Month,
   Branch,
   [Item Number],
   [PO Number],
   [Req Delv Date],
   [PO Amount],
   [Ordered Qty]
FROM Purchasing;
```

Most of the fields from each FACT table are not shared

How do I create a link table?

1. Create a key field with the common fields

2. Load all other fields with the key field from #1

3. Create a new table with the same key (link key) and the common fields separately ➔ Use DISTINCT

4. Repeat above for other tables

Link Table !!

5. If all the tables do not share the exact same fields, create separate keys for each table in the link table

How do I create a link table?

1. Create a key field with the common fields
2. Load all other fields

```
Sales:
Load
  Year & '_' & Month & '_' & Branch & '_' & [Item Number] as Key,
  Year,
  Month,
  [Branch],
  [Item Number],
  [Customer Number],
  [Invoice Number],
  [Order Number],
  [Salesman Number],
  [Invoice Date],
  [Sales Amount],
  [Sales Qty],
  [Cost Amount],
  [Margin Amount],
  [Unit of Measure]
FROM Sales;
```

How do I create a link table?

3. Create a new table with the same key and the common fields separately

```
LinkTable:
Load DISTINCT
  Year & '_' & Month & '_' & Branch & '_' & [Item Number] as Key,
  Year,
  Month,
  [Branch],
  [Item Number]
FROM Sales;
```

How do I create a link table?

- If all the tables do not share the exact same fields, create separate keys for each table in the link table

create separate keys for each table in the link table

```
Sales:
Load
   Year & '_' & Month & '_' & Branch & '_' & [Item Number] as SalesKey,
   [Customer Number],
   [Invoice Number],
   …
   [Margin Amount],
   [Unit of Measure]
FROM Sales;
```

```
LinkTable:
Load DISTINCT
   Year & '_' & Month & '_' & Branch & '_' & [Item Number] as SalesKey,
   Year,
   Month,
   [Branch],
   [Item Number]
FROM Sales;
```

How do I create a link table?

```
Sales:
Load
   Year & '_' & Month & '_' & Branch & '_' & [Item Number] as SalesKey,
   [Customer Number],
   [Invoice Number],
   …
   [Margin Amount],
   [Unit of Measure]
FROM Sales;
```

```
LinkTable:
Load DISTINCT
   Year & '_' & Month & '_' & Branch & '_' & [Item Number] as SalesKey,
   Branch & '_' & [Item Number] as InvKey,
   Year & '_' & Month & '_' & Branch & '_' & [Item Number] as POKey,
   [Branch],
   [Item Number]
FROM Sales;
```

```
LinkTable:
Load DISTINCT
   Year & '_' & Month & '_' & Branch & '_' & [Item Number] as SalesKey,
   Year,
   Month,
   Year,
   Month,
   [Branch],
   [Item Number]
FROM Sales;
```

How do I create a link table? - Final Scripts

```
Sales:
Load
  Year & '_' & Month & '_' & Branch & '_' & [Item Number] as SalesKey,
  [Customer Number],
  [Invoice Number],
  [Order Number],
  [Salesman Number],
  [Invoice Date],
  [Sales Amount],
  [Sales Qty],
  [Cost Amount],
  [Margin Amount],
  [Unit of Measure]
FROM Sales;

Inventory:
Load
  Branch & '_' & [Item Number] as InvKey,
  [On Hand Qty]
FROM Inventory;

Purchasing:
Load
  Year & '_' & Month & '_' & Branch & '_' & [Item Number] as POKey,
  [PO Number],
  [Req Delv Date],
  [PO Amount],
  [Ordered Qty]
FROM Sales;
```

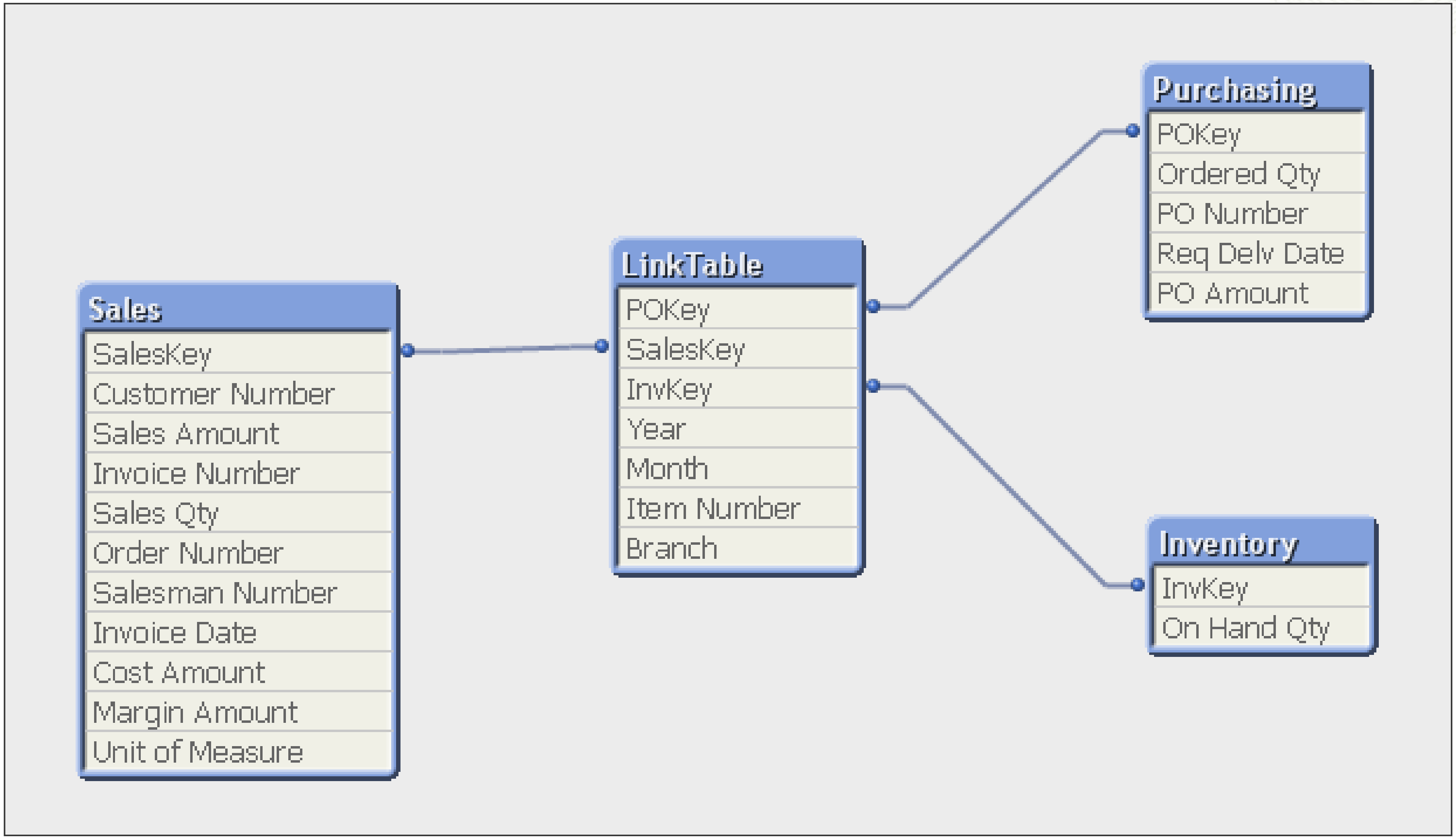
How do I create a link table? - Final Scripts

```
LinkTable:
Load DISTINCT
  Year & ‘_’ & Month & ‘_’ & Branch & ‘_’ & [Item Number] as SalesKey,
➔Branch & ‘_’ & [Item Number] as InvKey,
  Year & ‘_’ & Month & ‘_’ & Branch & ‘_’ & [Item Number] as POKey,
  Year,
  Month,
  [Branch],
  [Item Number]
FROM Sales;

LinkTable:
Load DISTINCT
➔Null() & ‘_’ & Null() & Branch & ‘_’ & [Item Number] as SalesKey,
➔Branch & ‘_’ & [Item Number] as InvKey,
➔Null() & ‘_’ & Null() & Branch & ‘_’ & [Item Number] as POKey,
➔Null() as Year,
➔Null() as Month,
  [Branch],
  [Item Number]
FROM Inventory;

LinkTable:
Load DISTINCT
  Year & ‘_’ & Month & ‘_’ & Branch & ‘_’ & [Item Number] as SalesKey,
➔Branch & ‘_’ & [Item Number] as InvKey,
  Year & ‘_’ & Month & ‘_’ & Branch & ‘_’ & [Item Number] as POKey,
  Year,
  Month,
  [Branch],
  [Item Number]
FROM Purchasing;
```

How do I create a link table? – Finished Result



Summary

Q: What is a link table?
A: It is a table that stores all possible combinations of values

Q: When do I use a link table?
A: When there is more than one field in common between tables

Q: What is the benefit?
A: To maintain integrity of your application

QlikView Challenge

synthetic key / join / link Table / concatenate

Performance / Usability

What do we mean by Data Model? - REVIEW

- “These sources and the underlying data will have to be manipulated within the script to deliver the Data Model that best suits your data for both **performance** and **usability**.”

Concatenate or Link Table?

Concatenated Models

- For **most scenarios Concatenation is the better solution**. It is **easy to manage**, **easy to extend** and takes **little development effort** to put in place.
- Concatenation comes **with two restrictions** to vet requirements against:

1. It does not cater for full transaction to transaction traceability.
   - i.e. I select SalesID, I won't see correlating Budget records. This is not strictly true, but it can be true in many scenarios and thus could be highlighted as a restriction.
2. It does not cater for implicit association between fact 1's unique dimensions and fact 2's transaction records.
   - i.e. If I select SalesCustomer, I won't see the Budget information that might be associated with the same year, month and product as the sales records filtered out.

Link Table Models

- Link tables replicate more traditional modelling, where a surrogate fact table (link) is put in place to resolve all associations between fact tables and common dimension tables.
- This might at first seem like a bullet proof solution to put in place every time – not true.
- The positive of link tables is that they resolve the relationships like any other table would. This gives full transaction traceability, even data implicitly associated via the other fact table is now traceable (select SalesCustomer – you will see the associated Budget records).

Link Table Models - Downsides

1. Inherently complex to build. Generating the link table yourself is no easy feat. There is considerably more sanity checking to be made to trust the code to produce the model.
2. The link table acts as a de-normalised table, meaning that representing high level associations like Budget at Month and Group level would require de-normalisation to the lowest common denominator with other facts, say Sales at Product and Date. This gives rise to a potentially large volume of links in the link table required to resolve Month and Group into correlating Dates and Products.

- The second downside is not exclusive to LinkTables – it is equally a challenge when concatenating fact tables together.

General Guidelines

- Star & Snow Flake schemas work best in QlikView. Relational tables tend to have loops (circular references) and therefore do not work correctly when brought into QlikView.

- The 4 main guidelines for modelling can be distilled as:

General Guidelines

1.

- Aim for a star schema. Flaking is ok, but try to keep it to a minimum as it may impact performance adversely to have too many tables hanging off tables.

General Guidelines

2.

- When de-normalising data (rolling up) in order to reduce flaking, stop if de-normalising means replicating records millions of times – the memory pointers required to store the same value enormous amounts of time now becomes significant.

General Guidelines

3.

- For multi-fact solutions, analyse requirements to see if a concatenated solution meets the needs. If transaction record traceability is crucial, rather than analysis through association of common dimensions, then look at whether a link table would suit. If neither model is a good fit, a custom data model must be delivered through careful consideration of requirements and iterative delivery. It may incorporate elements of both link and concatenated tables.

General Guidelines

4.

- In larger environments whether from a data volume, complexity or concurrency of user perspective, efficient QlikView document design become increasingly important. To this aim, please utilise the tools at your disposal regarding performance testing.

Conclusions…

A Word about Requirements

- Requirements will always inform your schema design.
- If you do not fully understand your requirements and these requirements are not thoroughly documented you are not ready to begin scripting. No exceptions.
- Requirements are focused in the problem domain; not the solution domain.
- Most Schema design questions are not really schema design questions they are really requirements questions.

Observations

- **There Is No One Best Architecture**.
- Architecture Is Entirely Dependent on Requirements
  - Systems, Skill Sets, Security, Functionality, Flexibility, Time, Money, and above all… **Business Requirements!**
- Likewise **Best Practices are not Universal**
- **Apply Best Practices on a per situation basis**
- **In Presales, getting the result more rapidly is often more important than over-thinking data modeling**
- **Remember that "Best Practice" will always be applied post-sale!**

Objectives Review

- Defining Data Models
- Understand how QlikView is Different from SQL
- Understand Data Warehousing Theory
- Adopt Applicable QlikView Data Modeling Best Practices

Questions