Advanced Topics in Dimensional Modeling

DAMA Philadelphia
January 8, 2008

Presented by
Tom Haughey, President
InfoModel LLC
868 Woodfield Road
Franklin Lakes, NJ 07417
USA
201 755 3350 ©
201 337 9094
tom.haughey@InfoModelUSA.com

What Is A Dimensional Model?

- A dimensional model is a model in which the data is structurally classified as fact or dimension.

- General characteristics:

  - Query oriented
  - Structured around data usage not business rules
  - Organized roughly into base facts and dimensions of those facts
  - Based on identification of key grains of data and on characteristics of those grains
  - Consisting usually of snapshot, conglomerated data
  - Looks to reduce the number and depth of joins

- Two general patterns-

  - **Star** schema in which all multi-leveled dimensions are flattened
  - **Snowflake** in which at least one multi-leveled dimension is kept separate

Star or Snowflake?

- First, some technologies require a snowflake and others require a star.
- Second, some queries naturally lend themselves to a breakdown into fact and dimension. Not all do. Where they do, a star is generally a better choice.
- Third, there are some business requirements that just cannot be represented in a star.
- Fourth, a snowflake should be used wherever you need greater flexibility in the interrelationship across dimension levels and components.
- Fifth, whether you collapse Header and Line Item into one fact table (and thereby reduce the structure to a star) should be based on tangible factors and by conformity to the dimensional pattern.
- Sixth, sometimes dimensional data changes at different rates or ways; in such more complex history situations, a snowflake can be better.
- Seventh, whether the star schema is more understandable than the snowflake is entire subjective and anything but a foregone conclusion.
- Eighth, some tests have revealed no difference between the performance of a star and snowflake, and sometimes the snowflake was slightly faster.
- It is unwise to pre-determine what is the best solution. A number of important factors come into play and these need to be considered.

Snowflake Caveat

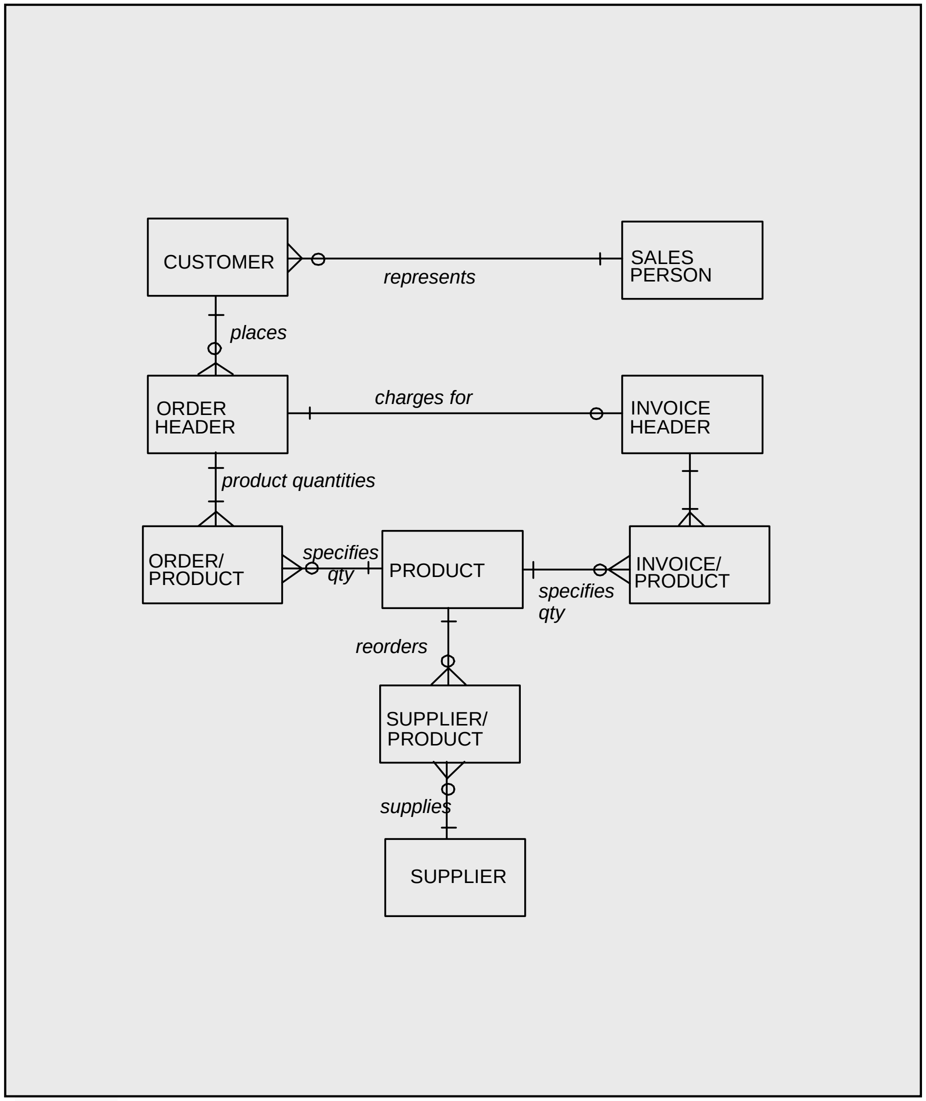
- Remember
  - Even if you choose to snowflake, it is not necessary to snowflake EVERYTHING
  - A hybrid is often practical
  - Because a Snowflake requires a join (say) up a hierarchy, it does not necessarily mean it is an I/O
  - DBMSs are smart and do three things:
    - Fetch data in pages, not rows
    - Read ahead
    - Buffer data that is frequently used
  - The most expensive DBMS operations are not joins but sorts and Cartesian products
    - The volume of data in the join is more significant than the presence of a join

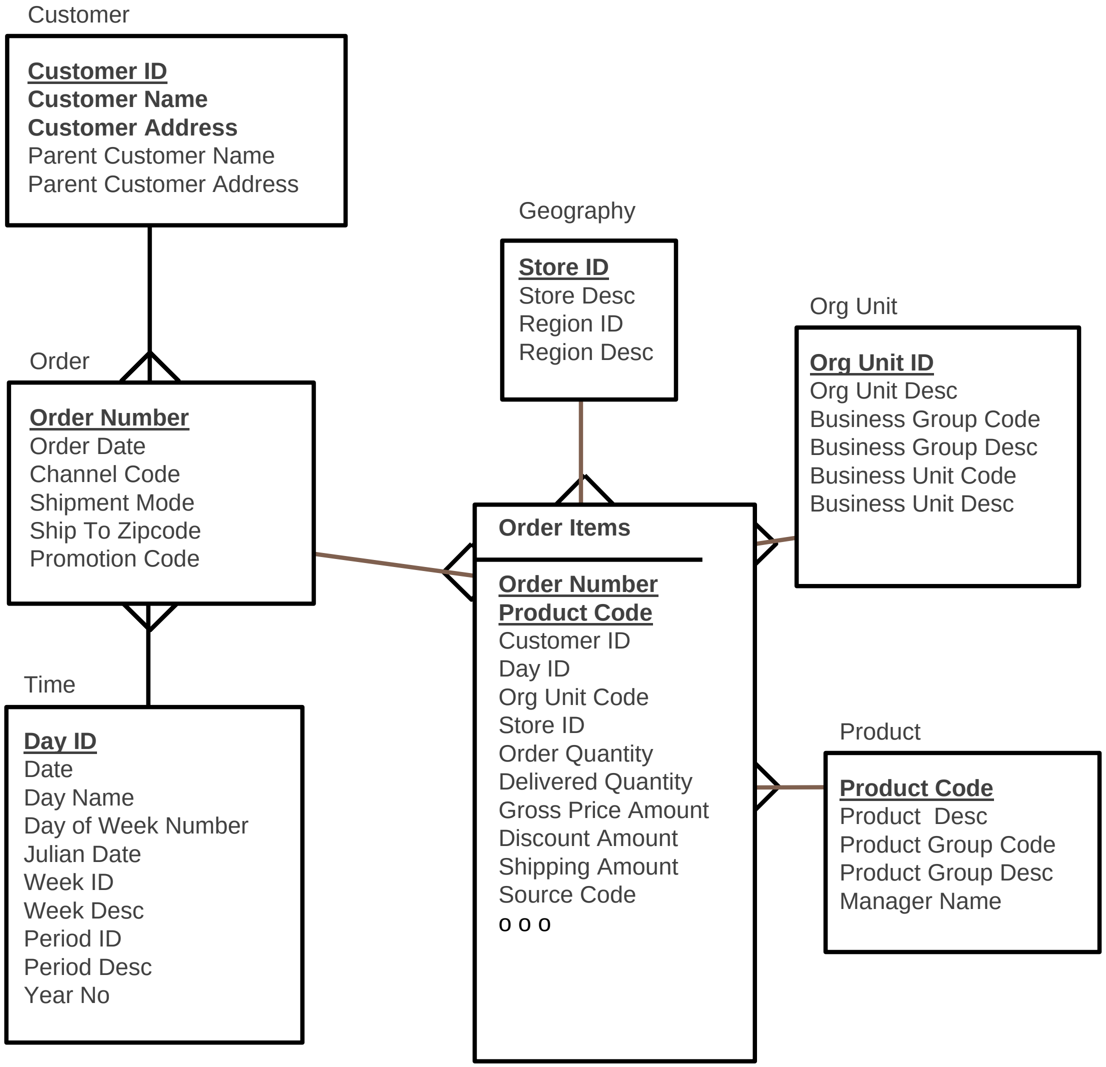
Observations on the Dimensional Model

- The following points represent the perspective of this presentation and are deviations from classical dimensional modeling theory:
  - Inherently, data is neither a fact or dimension. This is determined by query usage and should not be explicitly declared
  - Queries do not have to access from dimension to fact, or v.v.
    - It can be meaningful to query facts directly
    - It can be meaningful to query dimensions directly
  - Dimensions can be related to and join other dimensions without going through a fact
  - Facts can be joined to other facts
  - A foreign key in a fact can be null (in any valid form of null representation) if the relationship is optional
  - The choice of a design pattern (e.g., star) should be based on heuristics and not decided *a priori*
  - However, the capability to achieve these things can be limited by a given technology (e.g., MDDBMS* or Redbrick)

* MDDBMS = multidimensional DBMS

Operational vs. Analytical Model





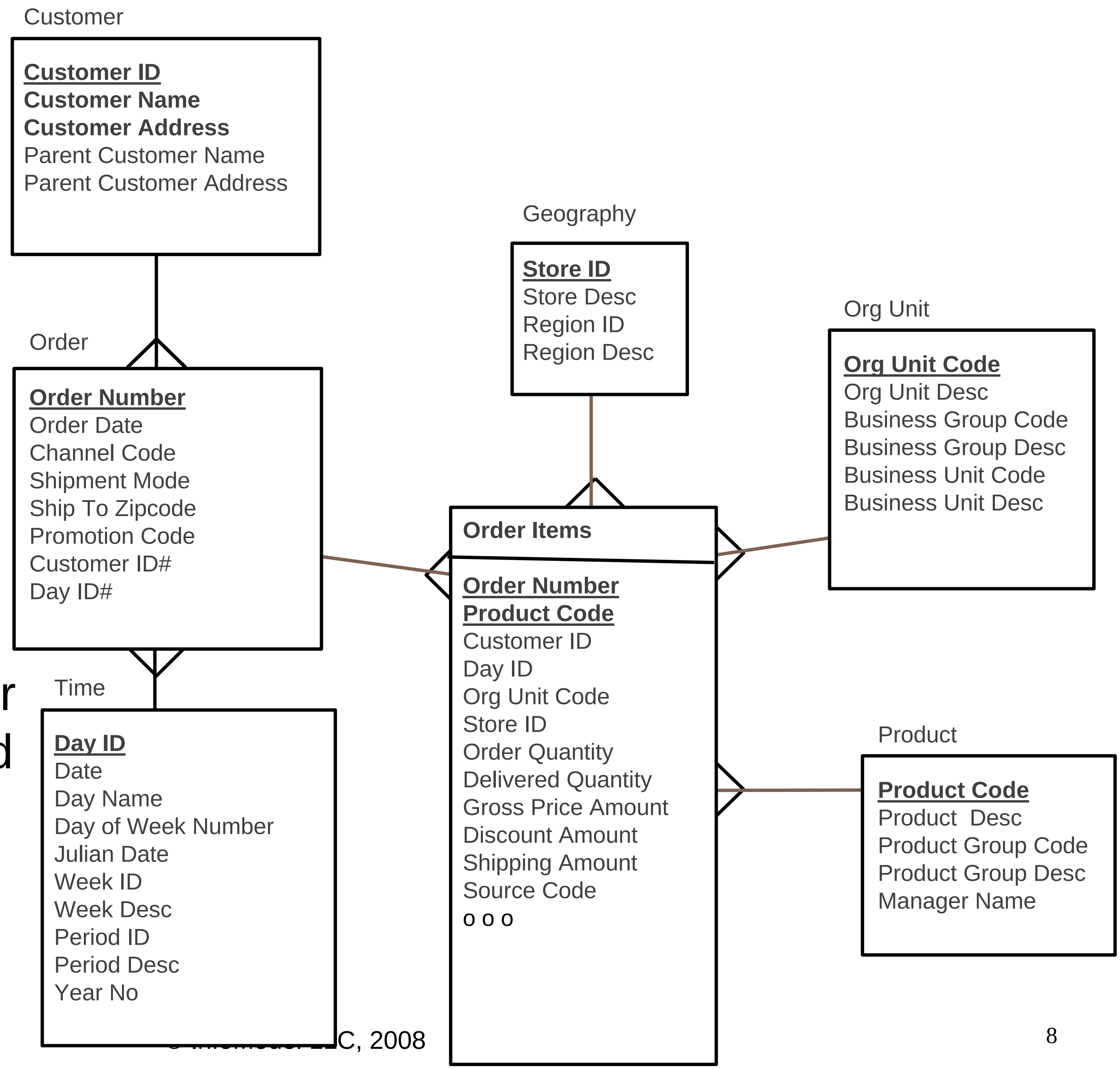
Operational Model **Analytical Model**

Normalization and the Dimensional Model

- Remember, if you take reporting or analytical data and normalize it, it will come out with a dimensional structure except for the following points:
  - Dimensions will be snowflaked
  - Compound facts (i.e., facts with multiple items with the transaction) will be divided into common data (the header) and the individual product sale (the line item)
  - For balance forward businesses, such as banking and insurance, facts will be divided into base facts (such as account or policy) and periodic status

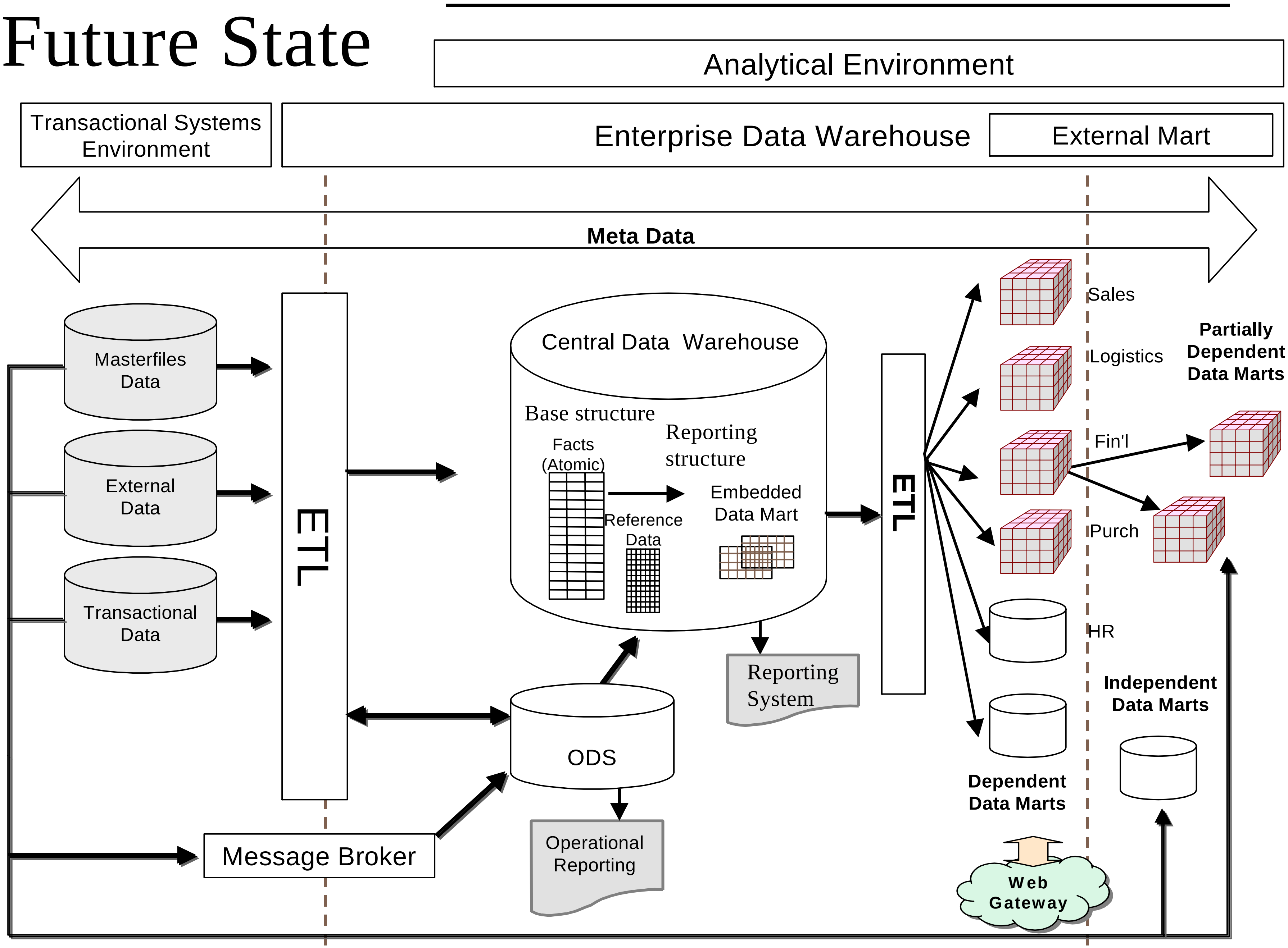
Complex Transactions (Line Item)

- Should you collapse Order Header?
- Consider the following factors:
  - Number of columns
  - Ratio of header to line item and
  - Data usage



The DW Environment

Future State



Overall Architecture of Data within the CDW

- **Base data**: DW data in its most atomic and flexible form; can be (and is) used for reporting
- **Reporting data**: data repackaged to facilitate reporting
  - **Detailed Reporting Data**: base data repackaged for reporting but of the same grain as base data, such as by flattening a recursive hierarchy
  - **Summary Reporting Data**: base data aggregated for reporting, derived from base data, or summarized in other ways

Does the CDW Model Have to be Dimensional?

- The central data warehouse database (CDW) is the heart of the DW
- The CDW model has to satisfy information requirements and must do so within performance expectations.
  - It must use whatever data compromises are necessary to achieve this
  - Many factors play into this
    - All of them tangible
    - None of them emotional (such as having to fit a pattern)
  - Because of the vast amounts of data in the data warehouse, we will see that there are many levels of data and potentially many levels of optimization
  - For the CDW, it is best to start with a logical model of the information requirements
    - Business oriented
    - Independent of technology
    - Independent of implementation

Functional Dependency

- Suppose technology were not an issue or that we had perfect technology
- The most flexible and open-ended model is one that honors functional dependencies (FD):
  - It shows the data with its most basic relationships
  - It shows the needs of the firm in its most basic form
  - FD's in operational data will be different than those in analytical data
    - What you are interested in will be different
    - The grains will be different
    - Business rules and rules-data will be different (such as pricing, discounting, etc.)
    - Either can be more or less detailed
    - Consider history in the DW
- The question is then:
  - Can your technology handle such a model?

Is DM Best Suited for "Fixed" Queries?

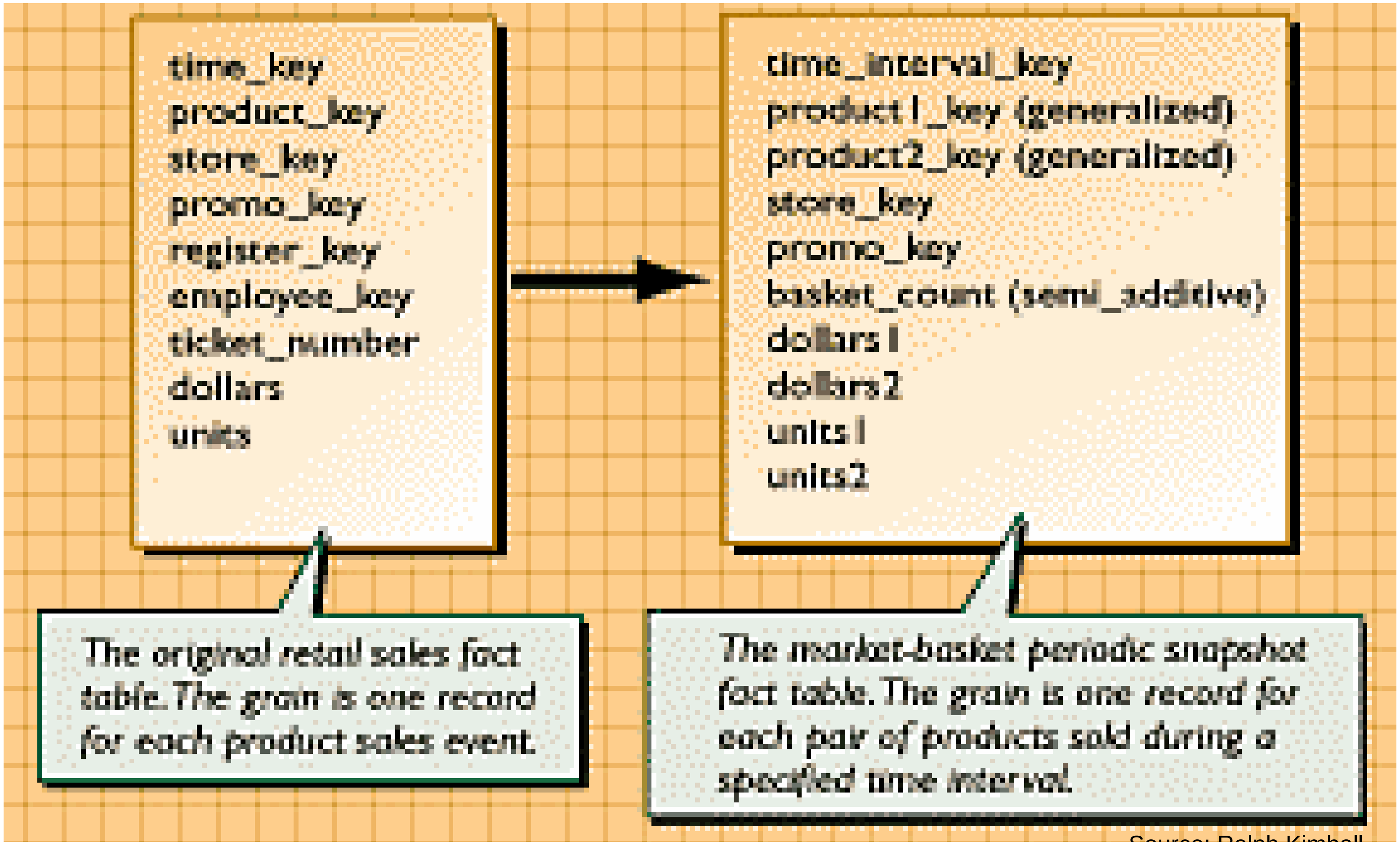
- Dimensional models (DM) can be used to support different types of queries and the queries do not have to be predetermined
- However, DM does rely on the distinction between a fact and a dimension and in that sense its queries are fixed
- Only certain types of queries lend themselves to this pattern
- The following do not:
  - What 2 products are sold together most often and when?
  - What 2 products were sold together most often in Florida last year during hurricane season?
  - What are the main characteristics of customers who lapsed last month?
  - What is the most profitable combination of investments for this type of investment customer?

Market Basket Analysis

- Identifies what combinations of products sell together
- Can easily be applied to other situations:
  - What Basic Cable purchase are followed by Sports Cable purchases
  - What customer characteristics typically occur together in a given situation, such as policy cancellation
- Seeking to understand what meaningful product combinations are sold together in individual market baskets
- Applicable to:
  - Collocating products within store displays
  - Separating frequently combined products
  - Packaging and pricing
  - Understanding combinations of brands
  - Looking for mixed aggregate results more meaningful
  - Understanding what does and does not sell well together

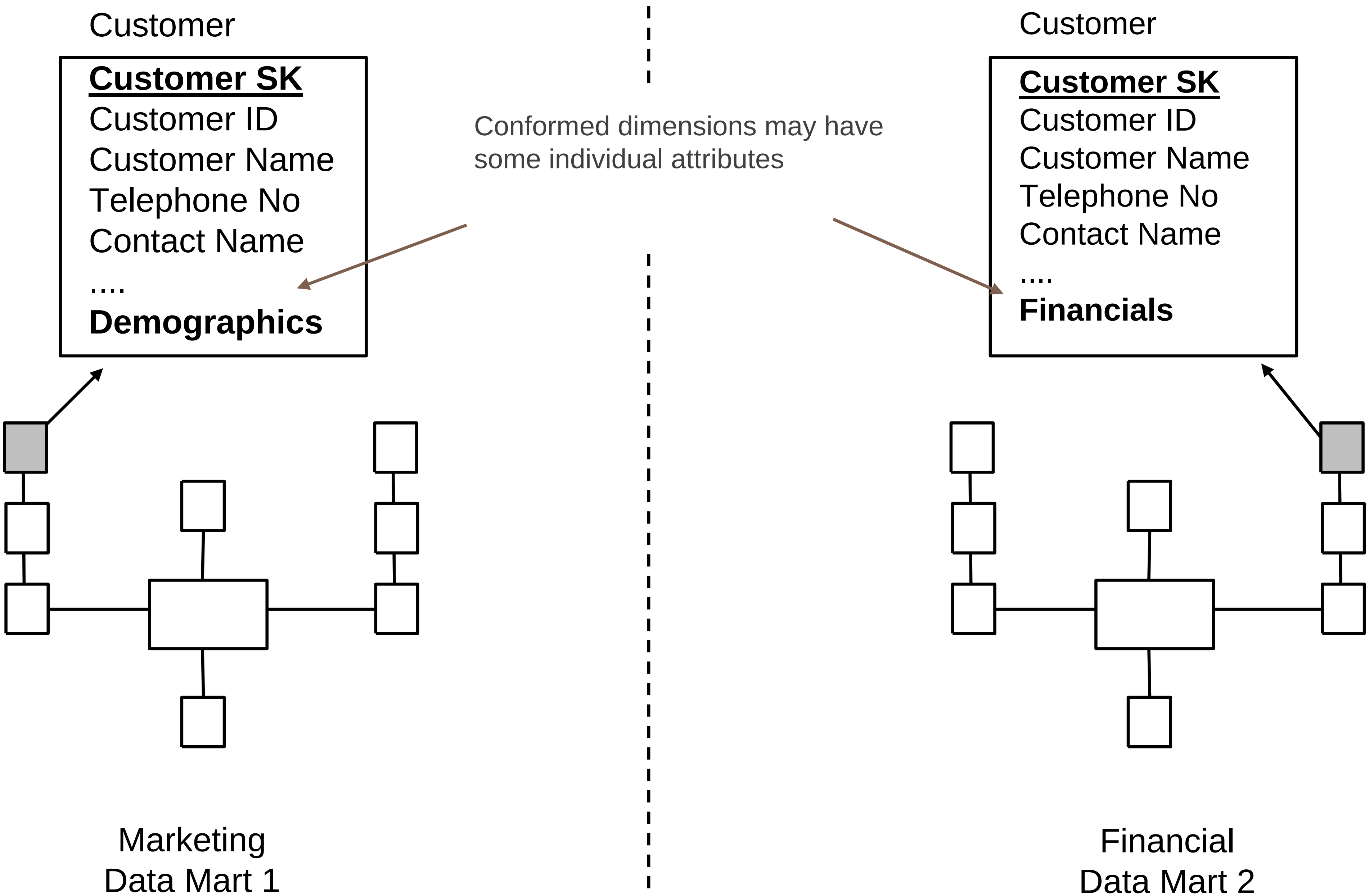
A Market Basket Example

- This example is well known but begs the question, meaning the statement presupposes the answer
- What do you do if you need the top ***4*** products, or in an insurance model, the top ***6*** characteristics of customers who lapsed last month? Or one time one thing, another time a another thing?



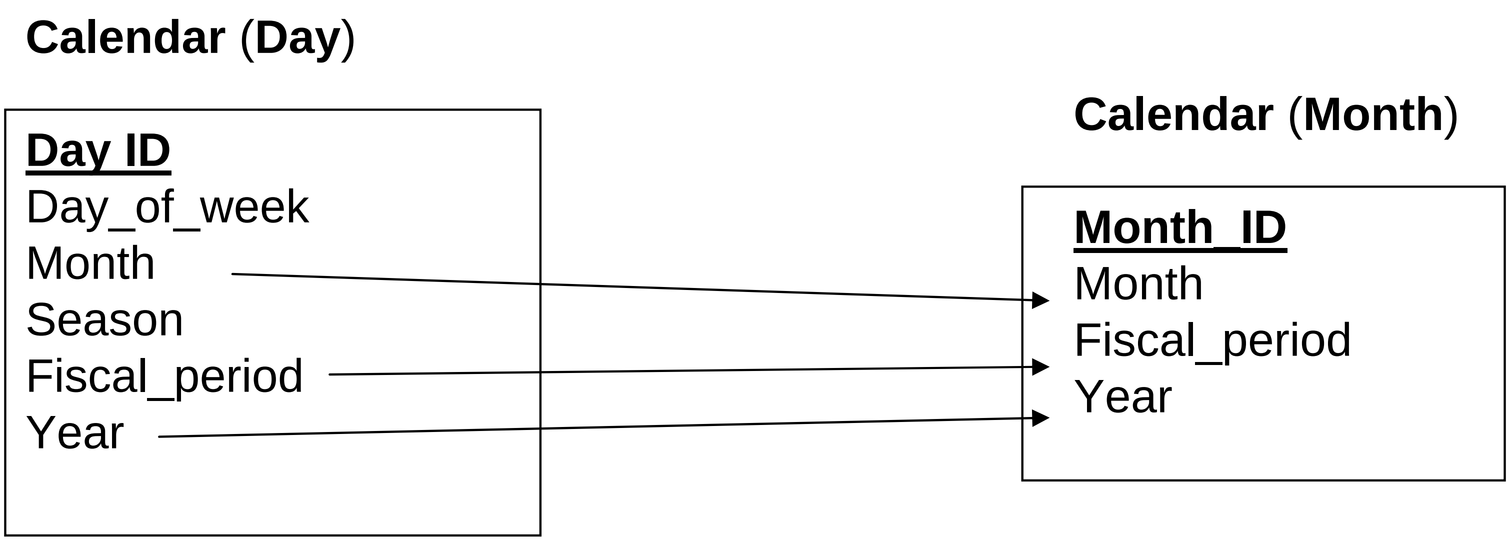
Source: Ralph Kimball

Conformed Dimensions: Equality Rule



Conformed Dimensions: Rollup Rule

- An attribute shared by multiple conformed dimensions must have the same business meaning and name so that it can be used as a common row header in separate queries
- Partially overlapping conformed dimensions are possible as the example shows:



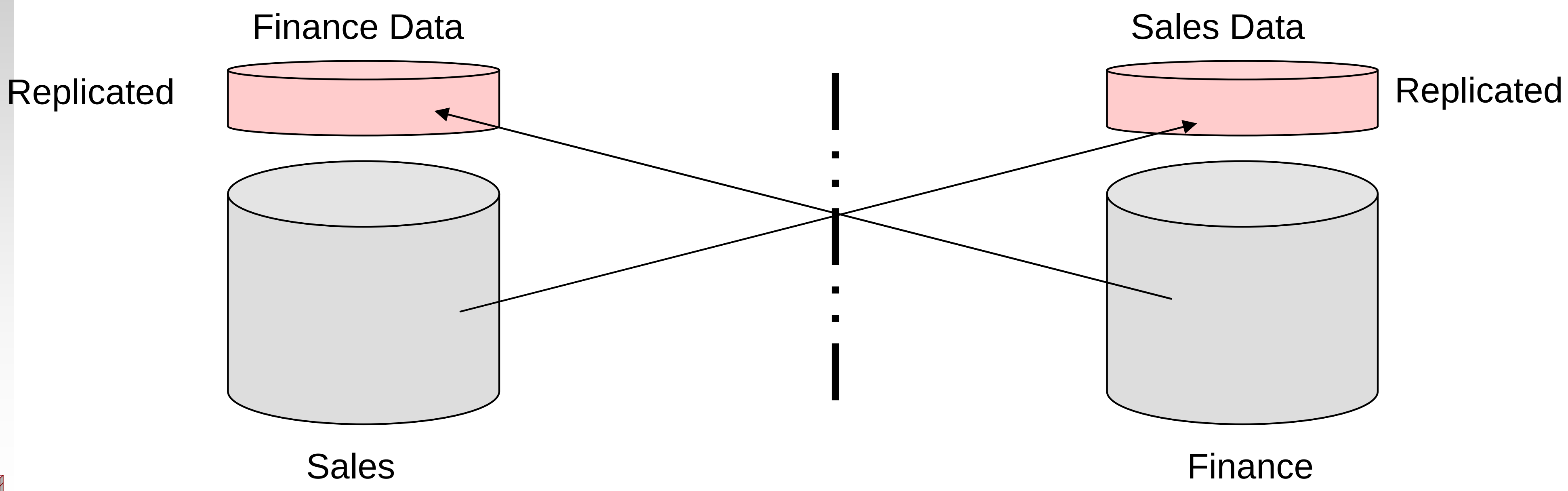
- All data marts must deploy the conformed dimensions simultaneously
- All aggregates affected by a dimensions change must be removed or modified

Conformed Dimensions

- Federation demands conformity
- Conformed dimensions means that when multiple copies of the same dimension exist, they are consistent with one another in that they honor either:
  - The equality rule
    - The same or compatible key
    - The same values for the key
    - The same grain or
  - The rollup rule
    - One is a strict rollup of the other
- Attributes in each conformed dimension can be a subset and can overlap
- Must be built at a common level of granularity (or be a rollup of the base dimension).
- Possibly, identified by a surrogate primary key.
- Conforming data in this way is productive, but it is difficult and costly
- Must establish an inviolable policy to reuse the conformed dimension whenever that subject is required for a data mart
  - Also called dependent dimensions or architected dimensions.

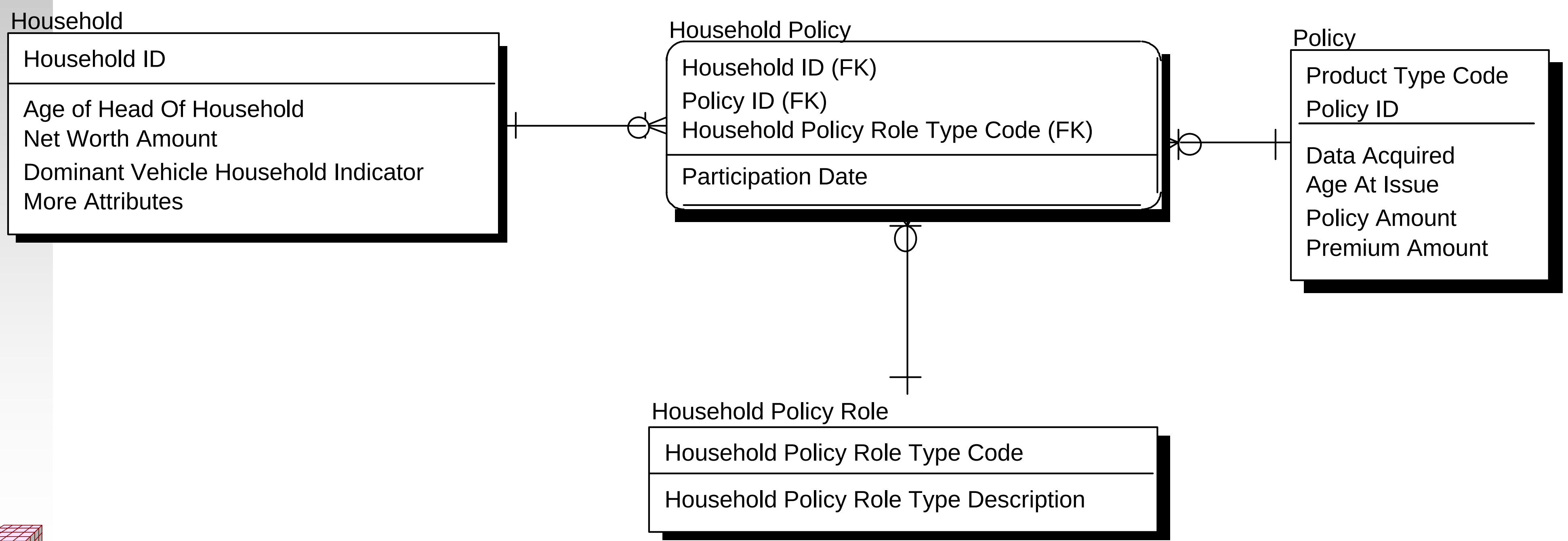
Crisscrossed Facts

- Conformed dimensions but crisscrossed facts
- When DWs are decentralized, whether they are conformed or not, there is a natural tendency for one mart to need the data contained in another
- The problem occurs because facts are not shared across marts



Snowflaked Many-to-Many Relationships

- Most dimension-to-fact relationships are one-to-many
- Some critical relationships can be many-to-many, such as Customer- or Household-to-Policy in insurance
- In the following example, assume Policy is the fact table

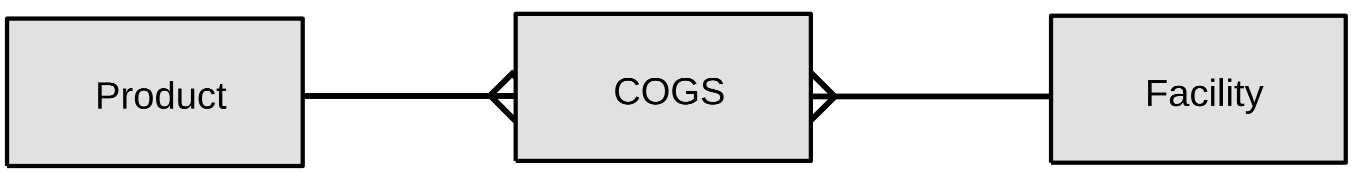


Typical Multi-valued Dimensions

- Multi-valued dimensions have a M : M relationship to the fact
- Sometimes supported with a weighting factor or allocation factor
  - Individuals, households in an account or policy
  - Diagnoses for a patient treatment
  - Industry classifications (SICs) for a company
  - Segment classifications for a household
  - Other (and different) examples:
    - Large customer dimensions
    - Financial product dimensions
    - Multinational and decentralized enterprise calendar dimensions
  - Report additive measures both correctly allocated and overlapping

Determining Facts or Dimensions

- Do all facts have to have a count or amount?
- Are all dimensions without them?
- Does it matter? Who cares?
- For example:
  - COGS (cost of goods sold – by product, time and location)
  - Factless facts
  - Give me the Customers that lapsed last month
  - Tell me which Facilities have Products with the lowest cost basis
  - Give me all Customers with Liquid Net Worth > $2MM



<u>Prod Code</u>
<u>Facility Code</u>
<u>Date</u>
Cost Amount

Periodic Status or Snapshot Requirements

- Often required in balance forward businesses such as banking and insurance
- Many transactions are not directly related to revenue
- Large number of transaction types, dimensionality and timing
- Revenue, status and other cumulative results needed at period end
- Achieved by the following:
  - Transaction time(s) rolled up to period end time
  - Transaction type dimension aggregated to periodic status dimension
  - Individual transaction amounts grouped and netted to multiple cumulative facts for period
  - Many other dimensions can remain the same

Periodic Status or Snapshot

Policy Transaction

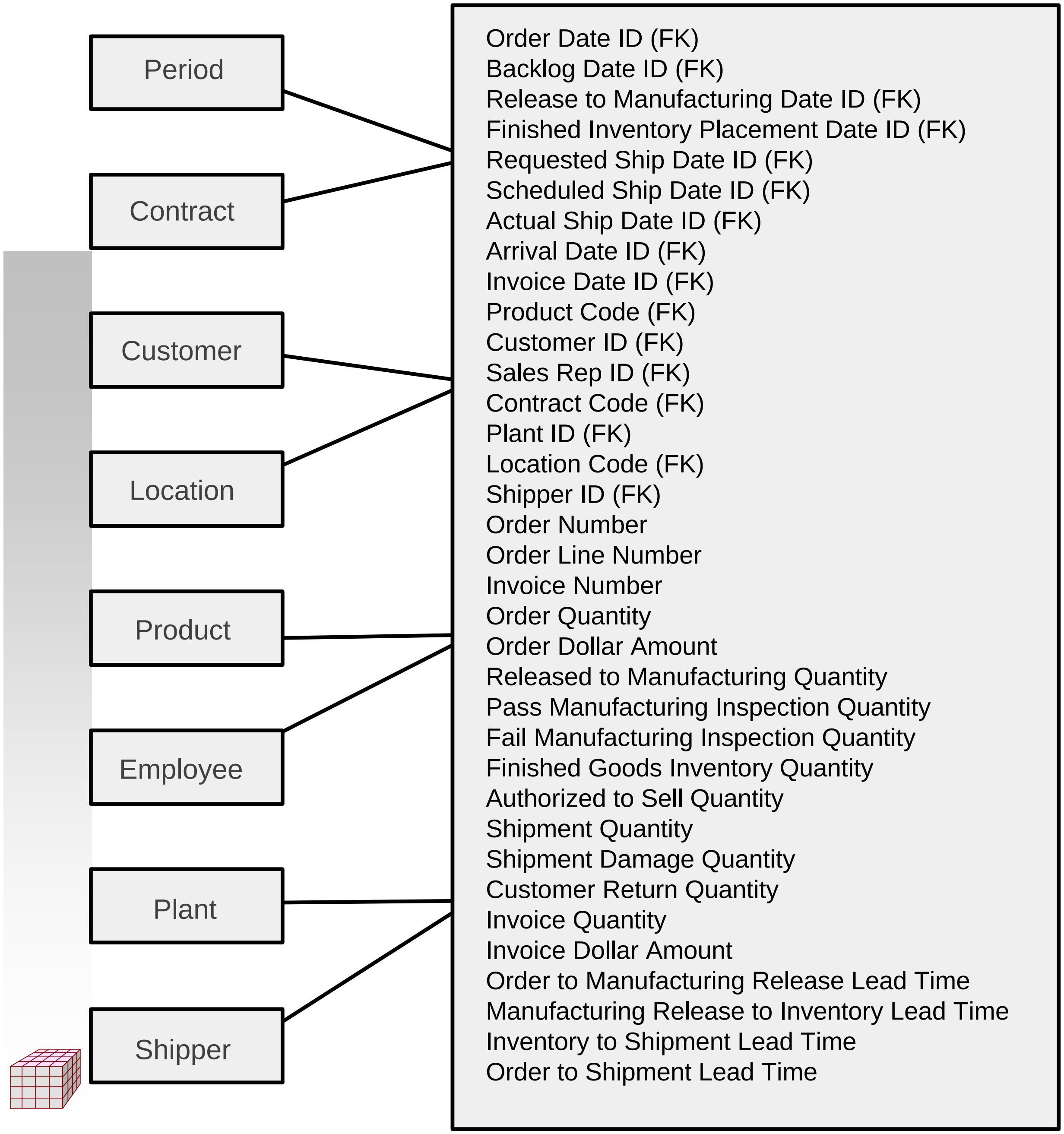
<u>Transaction ID</u>
Transaction DateTime ID
Policy ID
Product Type Code
Customer ID
Agent ID
Coverage Code
Covered Item Code
Transaction Type Code
Status Type Code
Transaction Amount

Periodic Status

<u>Period ID</u>
<u>Policy ID</u>
<u>Product Type Code</u>
<u>Policy Status Code</u>
Customer ID
Agent ID
Coverage Code
Covered Item Code
Status Type Code
Premium Paid Amount
Profitability Amount
Face Value Amount
Reserve Balance Amount
Transaction Count

Accumulating Snapshot

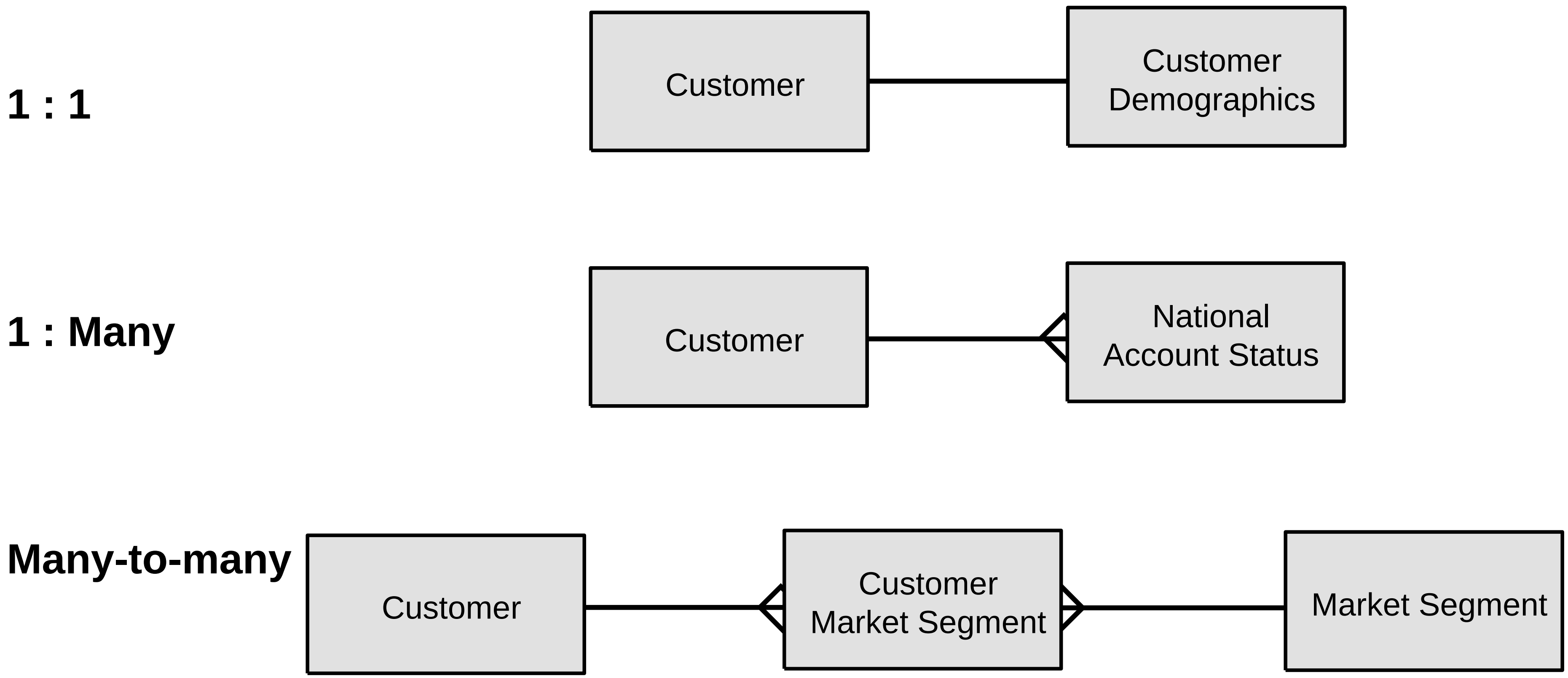
Order Processing Accumulating Snapshot



- In this accumulating snapshot, the complete timeline of a process is represented.
- Each fact consists of multiple groups of data, including multiple dates.
- Each repeating group represents one stage of the process.

- **Pros**
  - All the data is collected in one place for easy retrieval
- **Cons**
  - Limit as to how much data you can reasonably carry for each repeating group
  - Will work only if the relation of all stages is **1:1** to the order
  - The row must be updated at each milestone of the process
- **Alternative**
  - One row for each milestone.

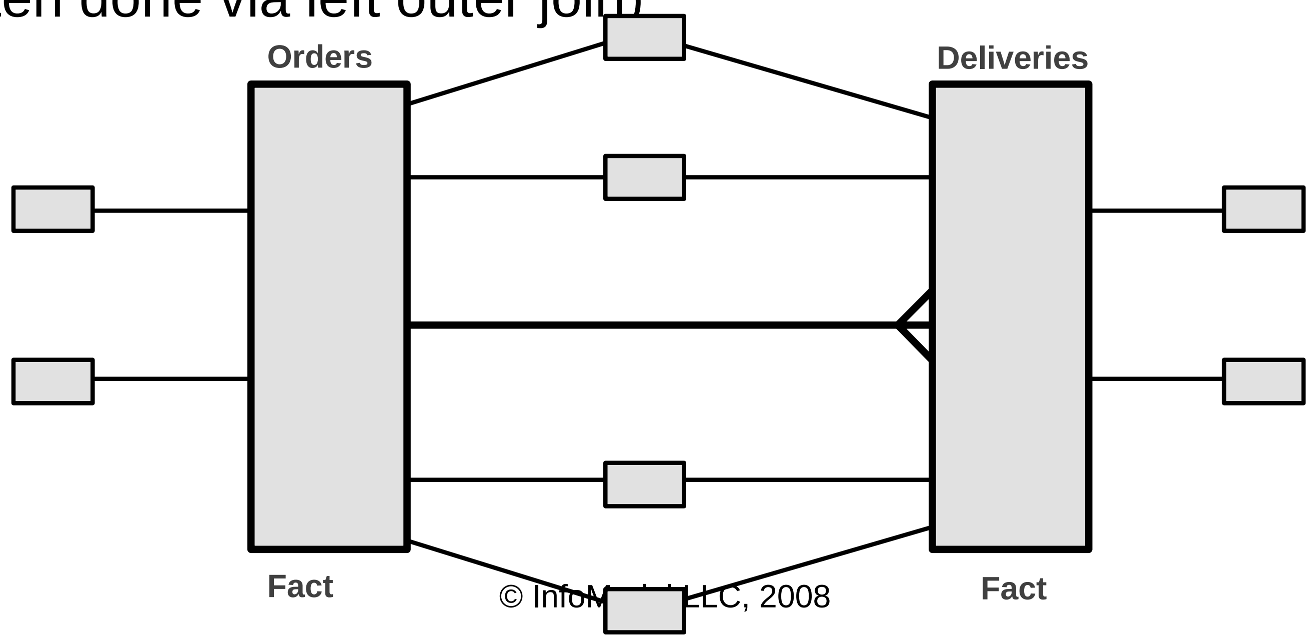
Dimension-to-Dimension Joins



- Dimension-to-fact relationships tell you what did happen
- Dimension-to-dimension relationships can tell you also what can happen
- Analyzing facts without dimensions is meaningless
- Analyzing dimensions, even without reference to facts, can be useful

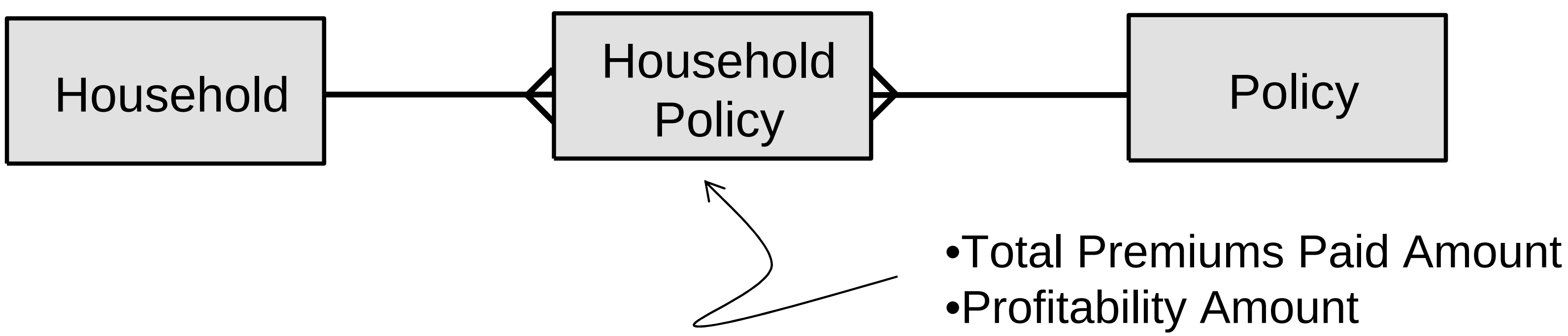
Fact-to-Fact Joins

- A relationship or join between two fact tables
- Considered taboo by dimensional modeling purists because:
  - It deviates from the dimension-fact paradigm
  - The table cannot be pre-classified as fact or dimension
  - Fact tables are voluminous requiring deep joins
  - Not supported by BI-specific technology especially:
    - Multidimensional DBMSs
    - Redbrick DBMS (only supports pure star schema)
  - However, data is data
- A robust DBMS platform will have no trouble accomplishing this join (often done via left outer join)



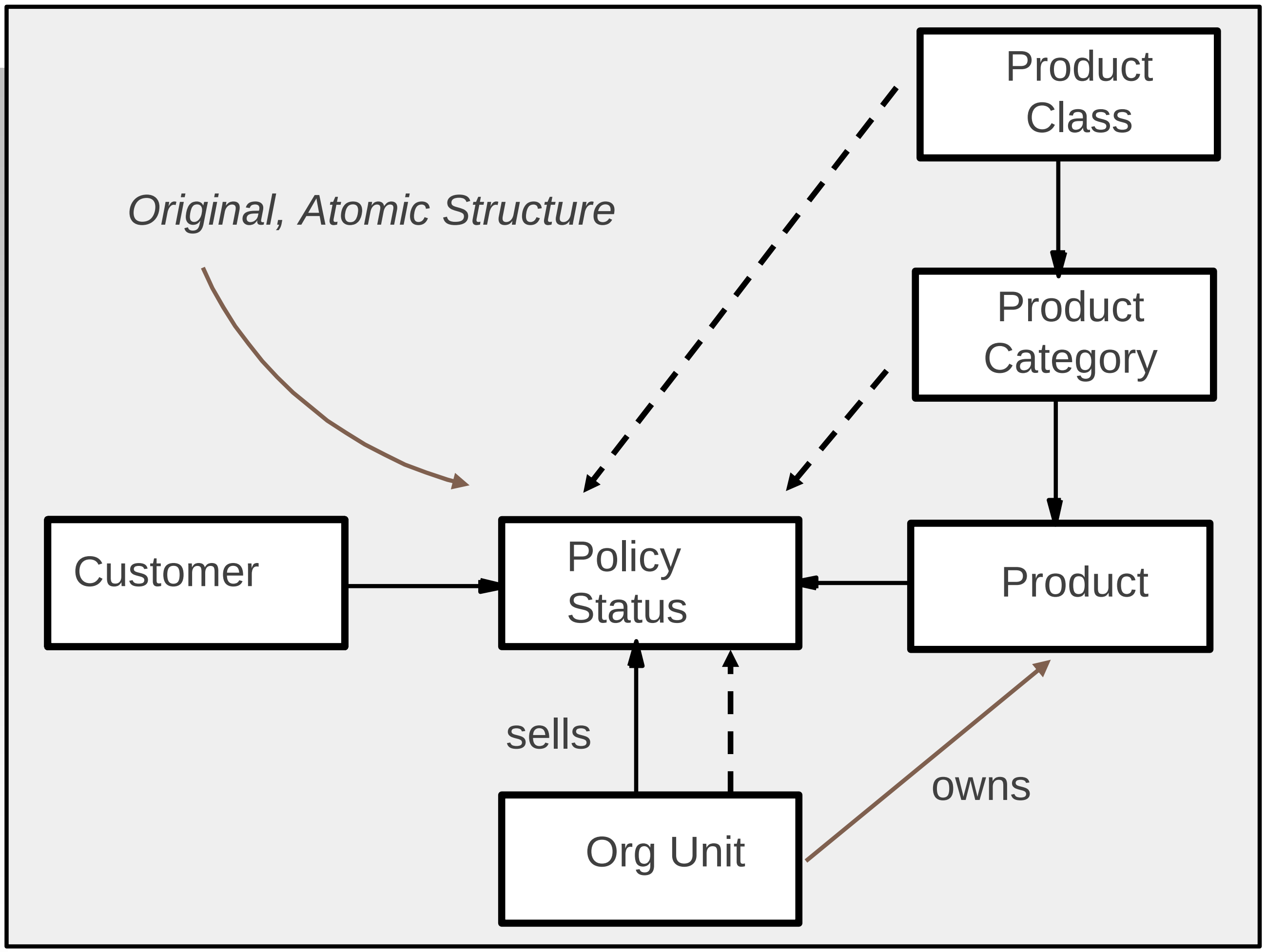
Factless Facts

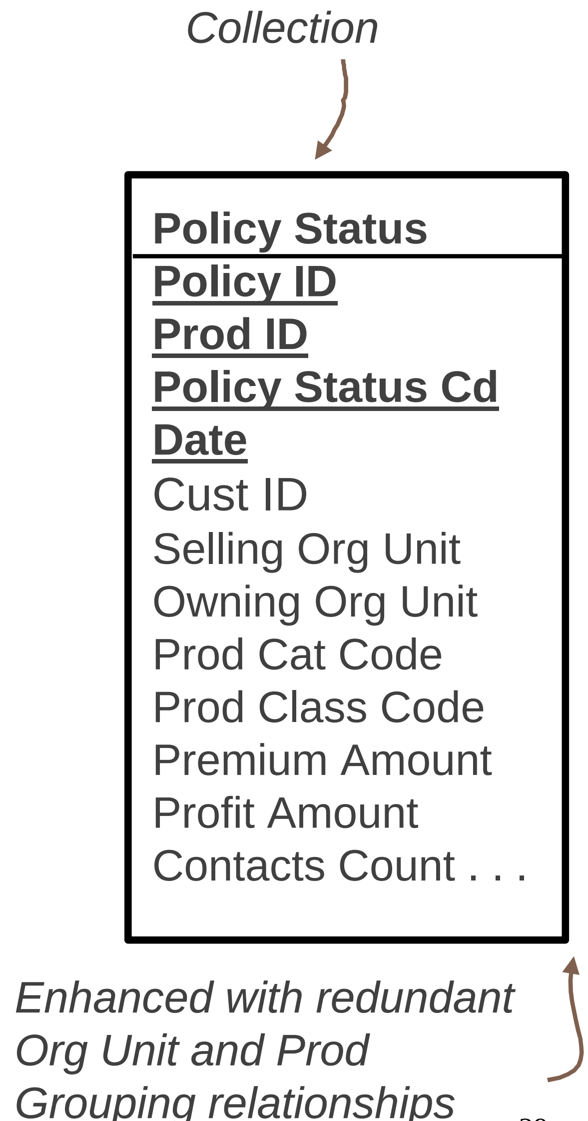
- Some “factless facts” may not be factless at all and could be expanded into collections of other data that can reasonable belong to it
- In effect, this is overloading the factless fact
  - – Household - Household Policy - Policy
  - – Customer - Customer Account - Account
    - Are Household Policy and Customer Account factless facts or complex dimensions? Before or after the changes suggested above?



Collection

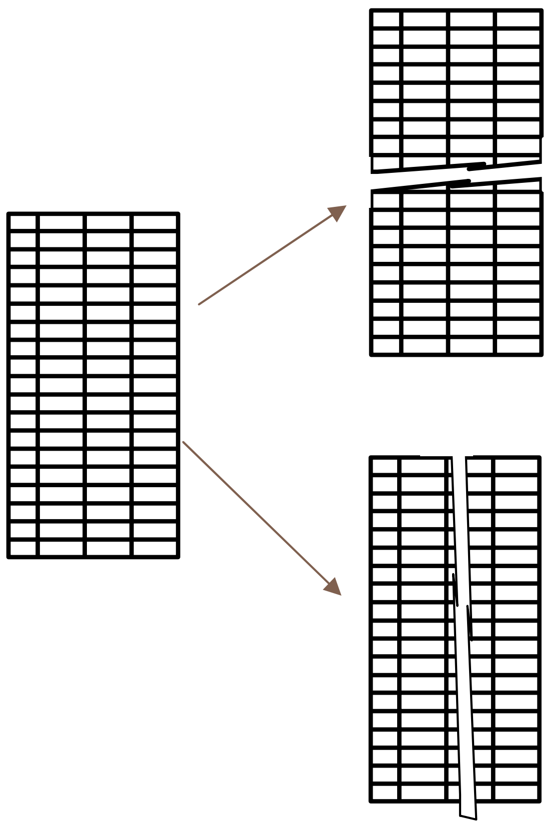
- Overloading a table by assembling into that table, the data or relationships that are often queried together, even though this may introduce some redundancy





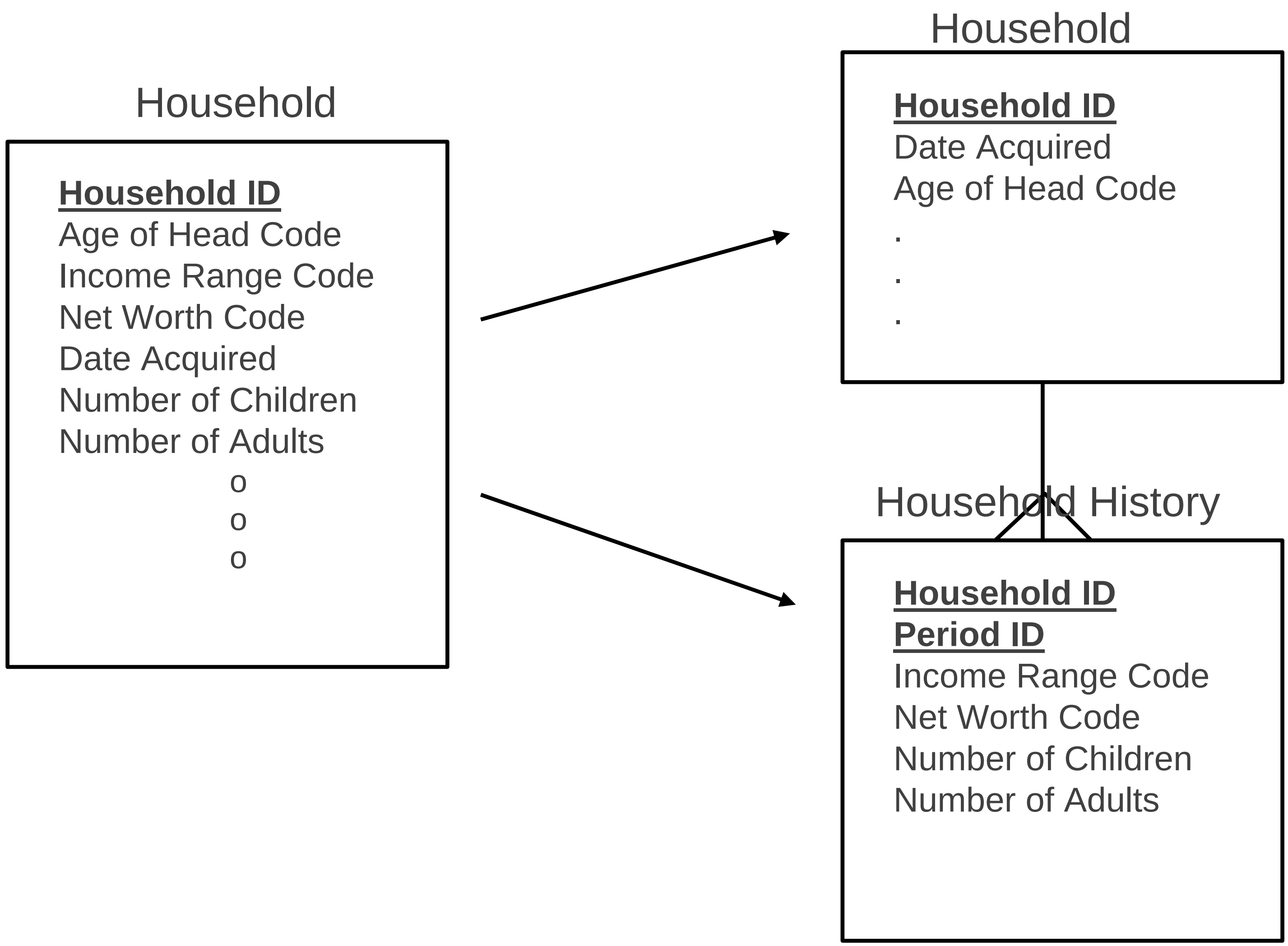
Very Large Dimensions (VLD)

- Not all Dimension Tables are small compared with the size of the fact table
  - ➔ Examples of very large dimensions:
    - Customer Dimensions in banking, insurance, telephone companies, catalog retailers
    - Household dimensions in any retail business (e.g., insurance, brokerage)
- Can hold millions of records
- Supporting complex analytical processing requires both careful modeling as well as the use of sophisticated DSS indexing and join techniques.



Very Large Dimensions (VLD)

- Split stable from changing data

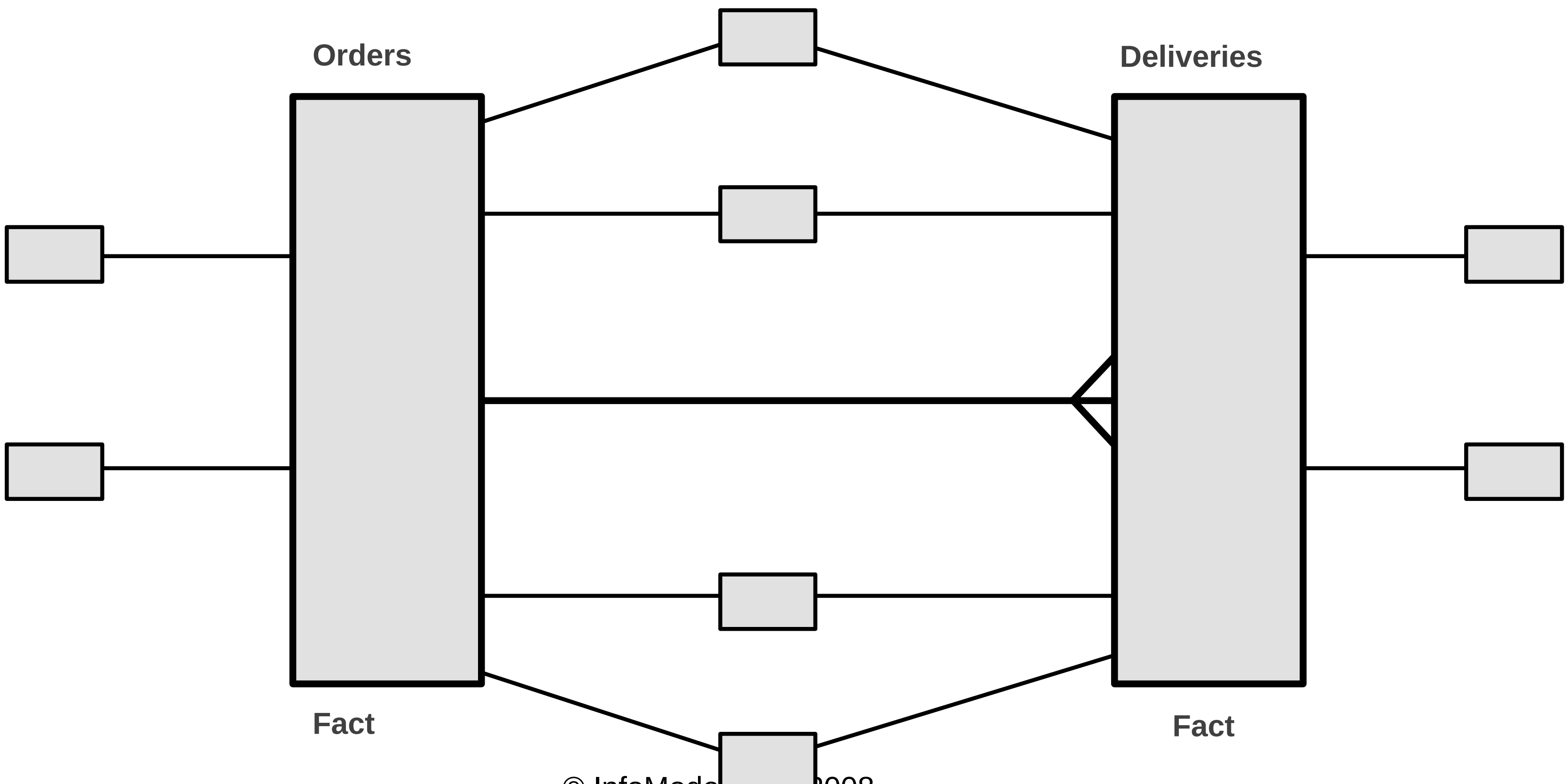


Heterogeneous Facts

- Heterogeneous facts are possible where the products offered or the markets (customers) addressed have different characteristics:
  - Heterogeneous customers:
    - ➔ Investment and insurance can have Private Clients and Capital Markets
  - Heterogeneous Products:
    - ➔ Again, insurance can have life, home, auto, etc., each with different attributes
    - ➔ Investment can have trades in stocks, options, etc., each with different attributes

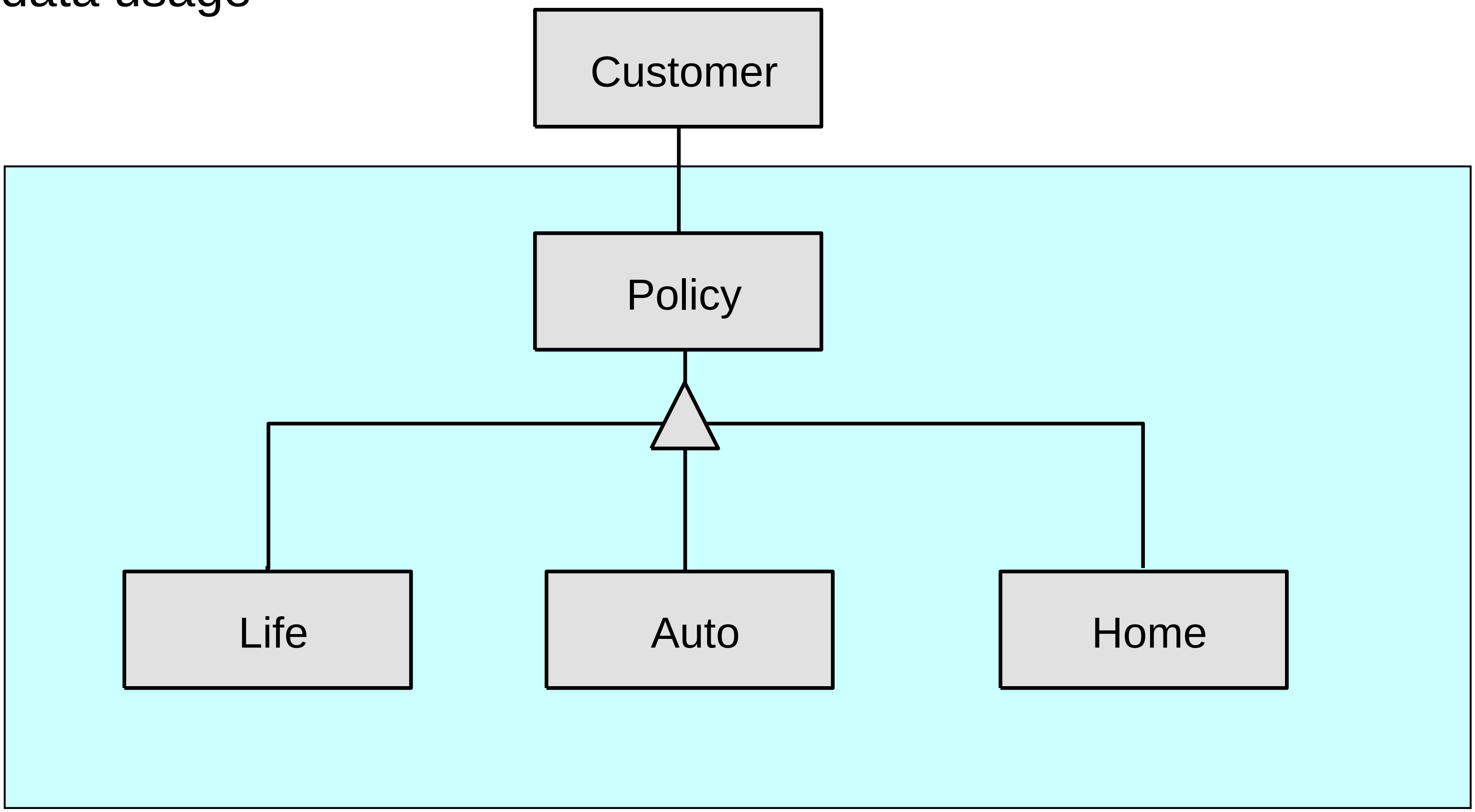
Purely Heterogeneous Facts

- If two facts are purely heterogeneous, such as, they have many different attributes or relationships, then they should simply be implemented as separate fact tables



Subtype Heterogeneous Facts in LDM

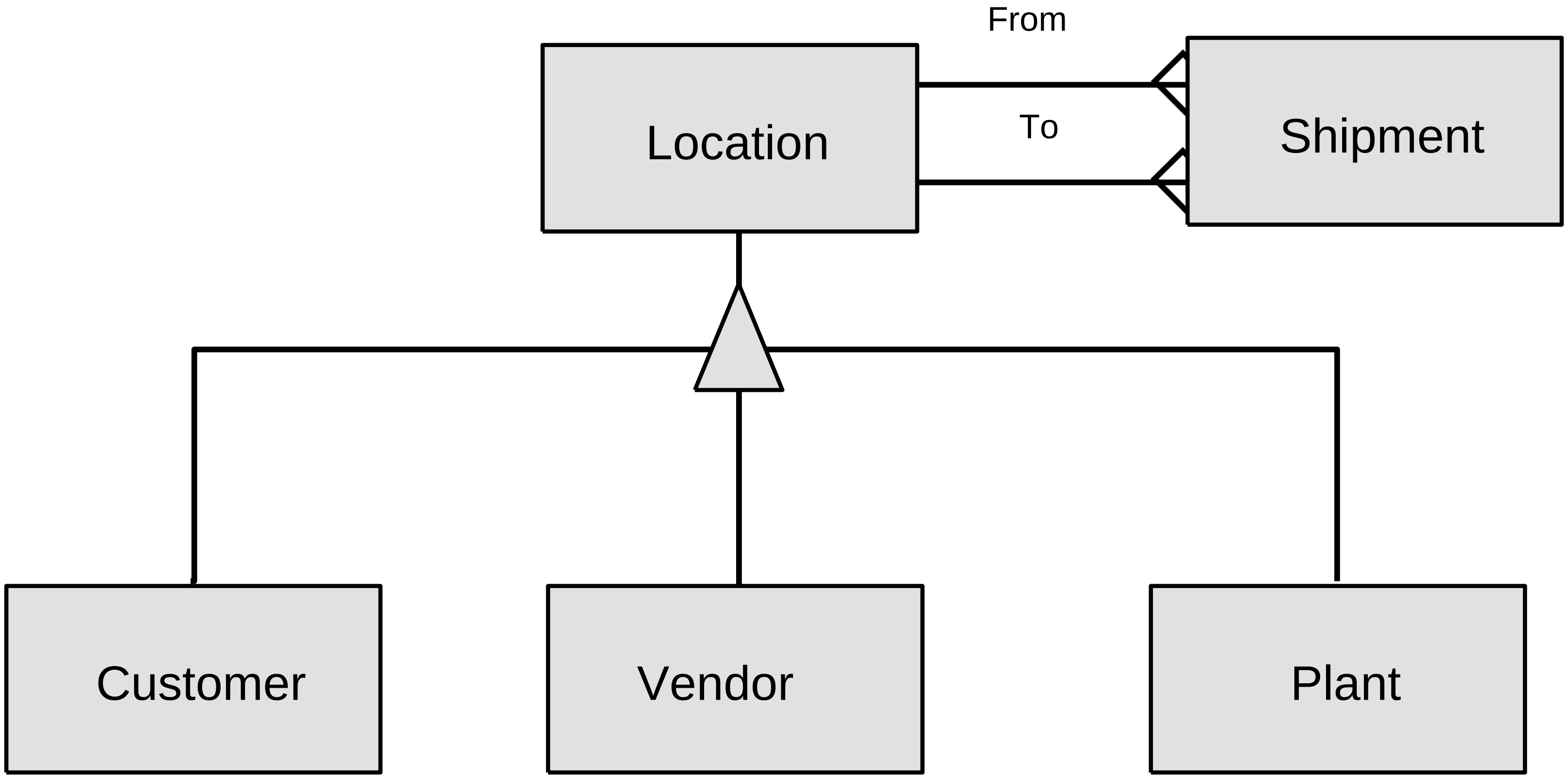
- When attributes or relationships are different, subtypes can be used
- Subtypes inherit all the characteristics of their supertype
- In implementing, consider number of columns, number of rows and data usage



* LDM = logical data model

Supertype Common Roles

- The so-called “Unity” dimension pulls together different specializations that have common roles.

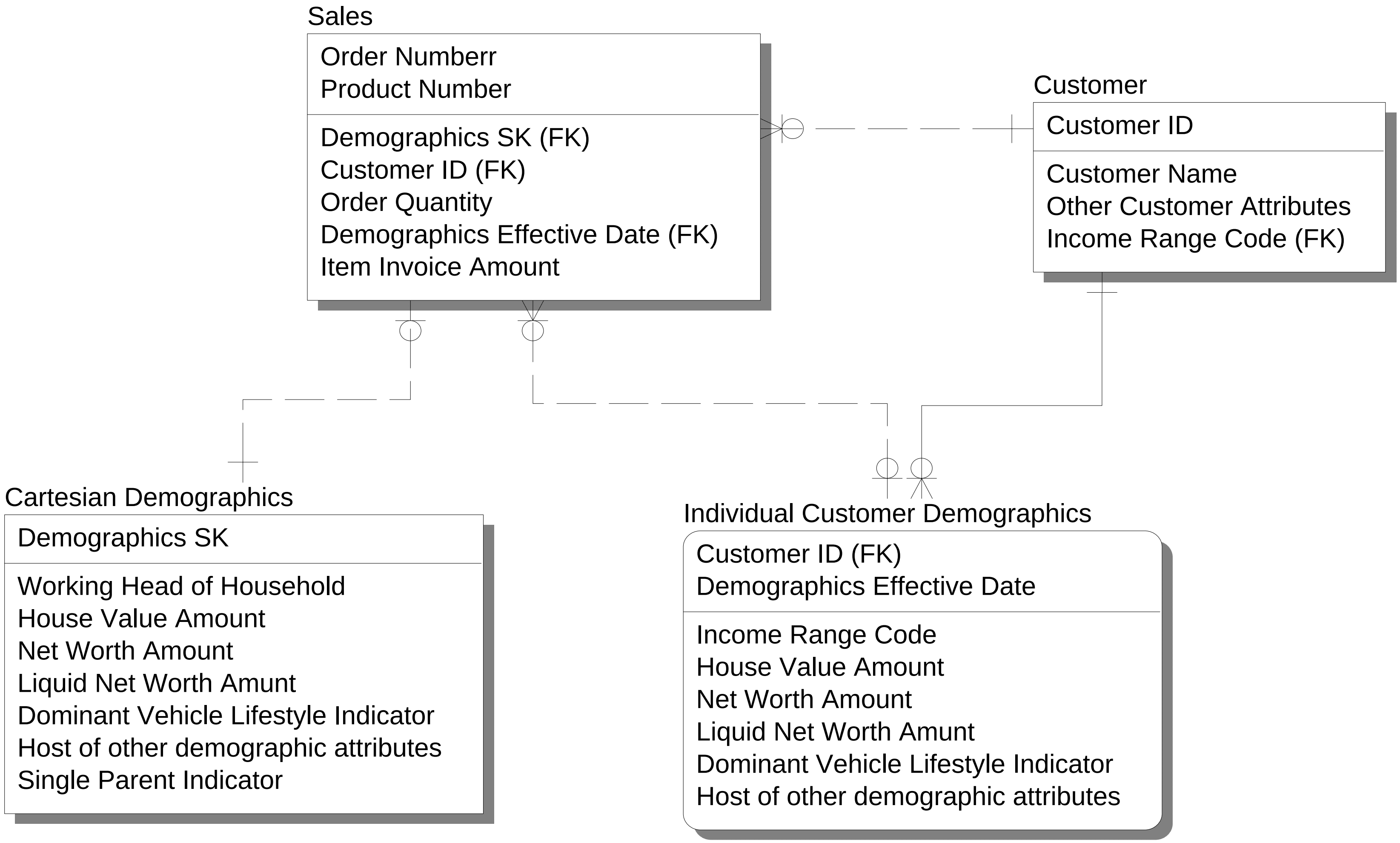


Cartesian Dimensions

- Cartesian product is formed by combining all instances of all participating dimensions
- A composite dimension that contains all permutations of related characteristics such as demographics
- Created in advance
  - Related to parent entity
  - The alternative is to create instances in the composite dimension as they occur in the fact
  - Cartesian demographic makes retention of history more difficult to do

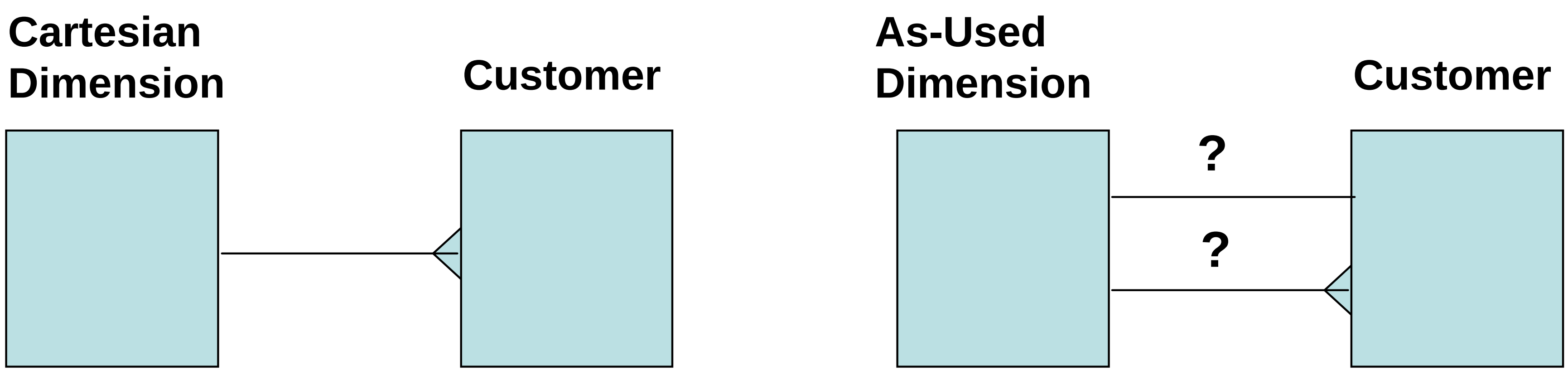
Cartesian Dimensions

- For illustration purposes, this example shows both kinds of demographic dimensions related to the same fact
- Note surrogate key for Cartesian, history for Individual Customer Dimension



Demographic Dimensions

- Can be constructed in two general ways:
  - Cartesian dimension: a dimension pre-populated with all possible combinations of demographic attributes
    - Can tell which combinations are not in use
  - Targeted dimensions: populated only when actual combinations exist
    - Cannot easily tell which combinations are not in use

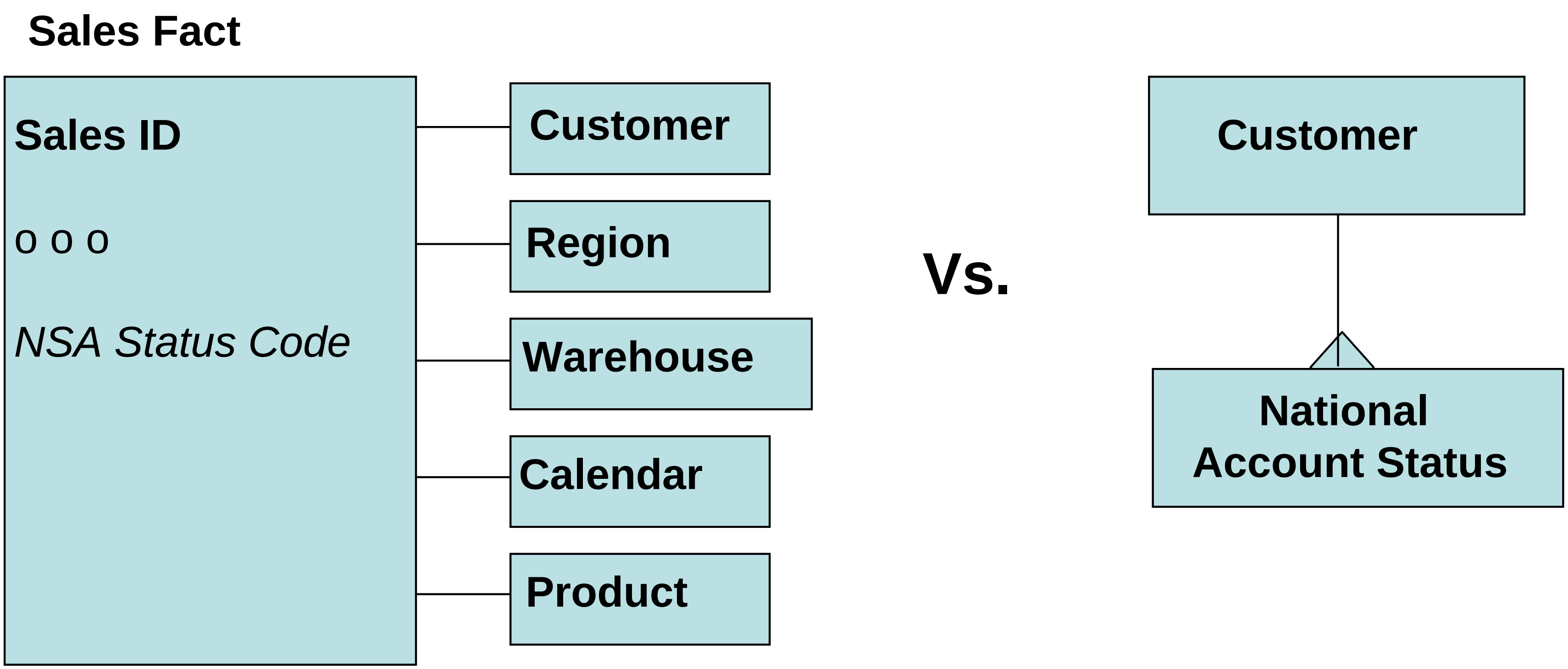


Rapid Changes

- Data models should always FIRST represent the data by enforcing functional dependencies and THEN evaluate the need to deviate from that
- To store a dimensional attribute directly in the fact table should be done:
  - If the dimensions changes at the same rate as the fact occurs
  - If the history of the dimension is not preserved some other way such as via a dimension history

Rapid Changes

- Conventional dimensional advice on rapidly changing dimensions (non-textual) is to put them in the fact table
- Which of the following is better for National Account Status (NAS)?



- Be sure to examine overall data usage (of different types)
- Perhaps it is better to include NAS in both fact (to tell which sales were national account sales) and dimension (to tell which accounts are national accounts)

Junk Dimensions

- A catch-all grouping of miscellaneous flags and indicators.
- Helpful, but not absolutely required, if a strong interrelationship among a group of miscellaneous dimensional attributes
- Used to reduce dimension clutter
- Collect together miscellaneous flags and indicators, and non-additive fact characteristics
- If possible, group such data that has a relationship
- Look throughout requirements for these associations
- Remove textual and unstructured data (such as comments) to its separate dimension

Junk Dimension

- Benefits:
  - A useful location for related codes, indicators and their descriptors
  - Simplify a design that already has many dimensions.
  - Provide a smaller, quicker point of entry for queries
  - Capture the context of a specific transaction.
- Insurance example:
  - Capture the context surrounding claims.
  - Claims, even similar claims, may be handled differently.
  - E.g., how the claim was reported, investigated and paid
- Two approaches for creating junk dimensions:
  - Create in advance (if few and well known) . Each possible combination represents one row.
  - Create during ETL as instances are encountered (if many)

Further Questions?

Finis