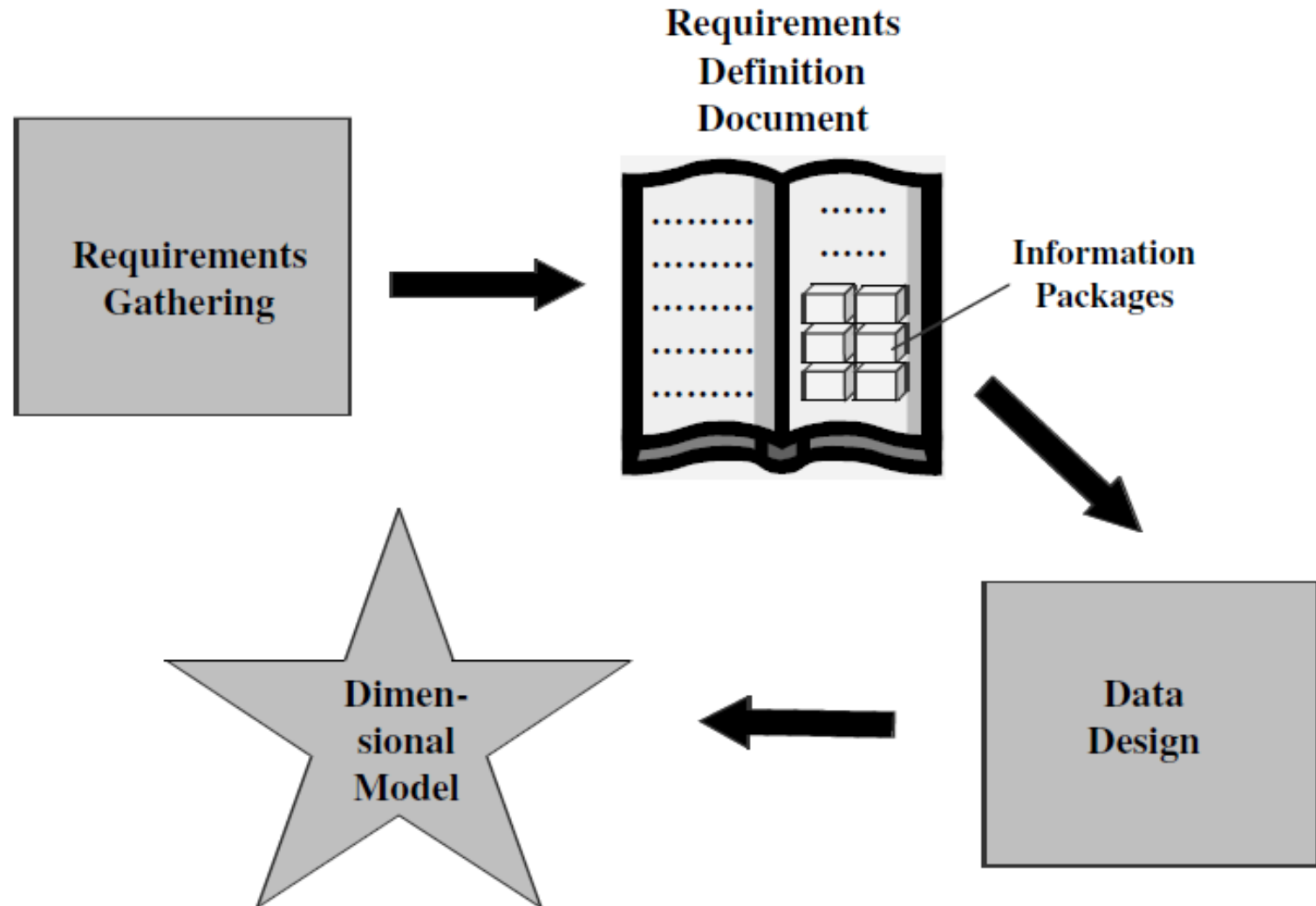


Unit-9

Dimensional Modeling

Requirements to Design



Design decisions to be taken

- Choosing the process:-deciding subjects
- Choosing the grain
- Identifying and confirming dimensions
- Choosing the facts
- Choosing the duration of the database

Dimensional modeling basics

Time	Product	Payment Method	Customer Demographics	Dealer	
Year	Model Name	Finance Type	Age	Dealer Name	
Quarter	Model Year	Term (Months)	Gender	City	
Month	Package Styling	Interest Rate	Income Range	State	
Date	Product Line	Agent	Marital Status	Single Brand Flag	
Day of Week	Product Category		Household Size	Date First Operation	
Day of Month	Exterior Color		Vehicles Owned		
Season	Interior Color		Home Value		
Holiday Flag	First Year		Own or Rent		
Facts: Actual Sale Price, MSRP Sale Price, Options Price, Full Price, Dealer Add-ons, Dealer Credits, Dealer Invoice, Down Payment, Proceeds, Finance					

Formation of the automaker sales fact table

Automaker Sales

Fact Table

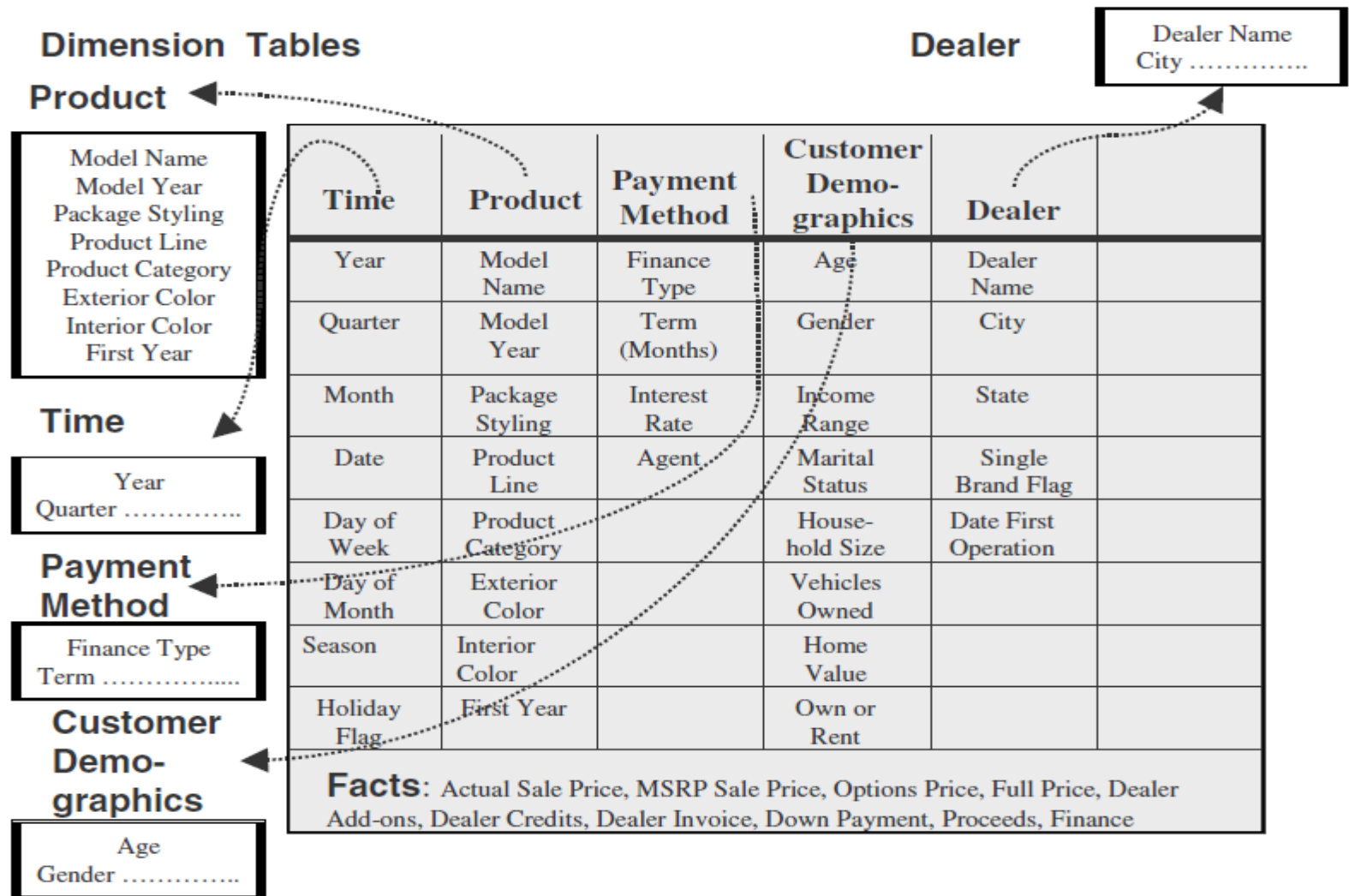
Actual Sale Price
MSRP Sale Price
Options Price
Full Price
Dealer Add-ons
Dealer Credits
Dealer Invoice
Down Payment
Proceeds
Finance



Dimensions

Time	Product	Payment Method	Customer Demo-graphics	Dealer	
Year	Model Name	Finance Type	Age	Dealer Name	
Quarter	Model Year	Term (Months)	Gender	City	
Month	Package Styling	Interest Rate	Income Range	State	
Date	Product Line	Agent	Marital Status	Single Brand Flag	
Day of Week	Product Category		Household Size	Date First Operation	
Day of Month	Exterior Color		Vehicles Owned		
Season	Interior Color		Home Value		
Holiday Flag	First Year		Own or Rent		
Facts: Actual Sale Price, MSRP Sale Price, Options Price, Full Price, Dealer Add-ons, Dealer Credits, Dealer Invoice, Down Payment, Proceeds, Finance					

Formation of the automaker dimension tables



How much sales proceeds did the jeep tata mahindra, 2005 model with vxi options, generate in january 2000 at spectra auto dealership for buyers who owned their homes, financed by icici prudential financing?

ER Model v/s Dimension Model

- ER diagram is a complex diagram, used to represent multiple processes. A single ER diagram can be broken down into several DM diagrams.
- In DM, we prefer keeping the tables de-normalized, whereas in a ER diagram, our main aim is to remove redundancy
- ER model is designed to express microscopic relationships between elements. DM captures the business measures
- DM is designed to answer queries on business process, whereas the ER model is designed to record the business processes via their transactions.

Entity-Relationship vs. Dimensional Models

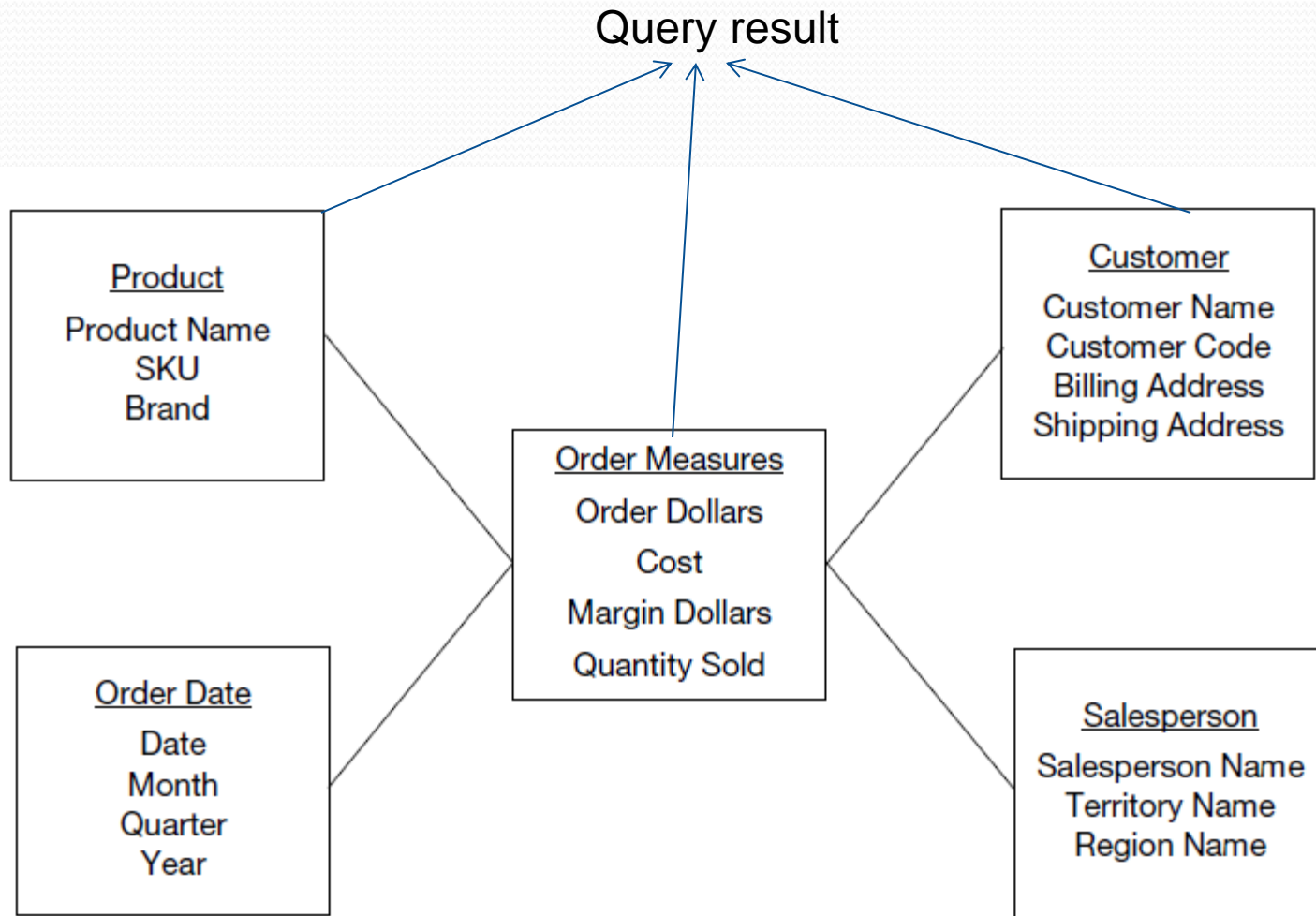
E-R DIAGRAM

- One table per entity
- Minimize data redundancy
- Optimize update
- The Transaction Processing Model

DIMENSIONAL MODEL

- One fact table for data organization
- Maximize understandability
- Optimized for retrieval
- The data warehousing model

Star Schema-example of order analysis



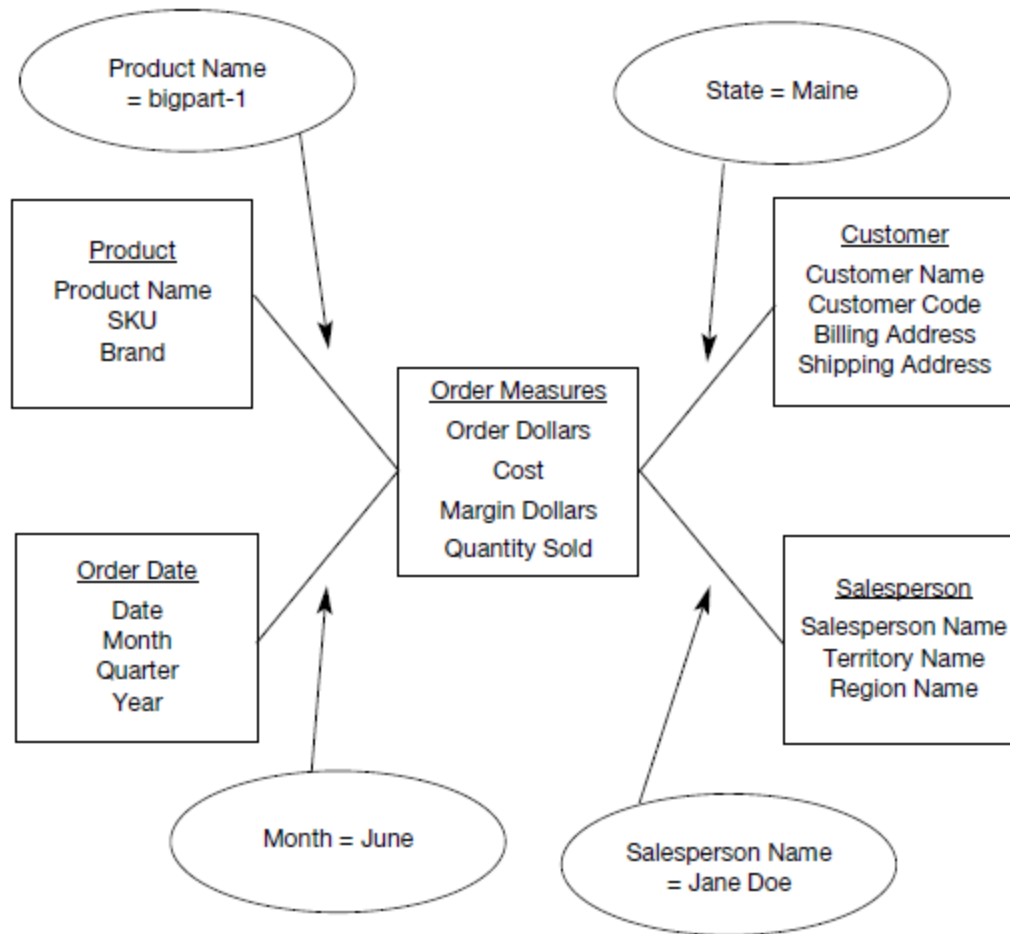
Star Schema-example of order analysis

- For a given amount of dollars, what was the product sold? Who was the customer? Which salesperson brought the order? When was the order placed?
- When a query is made against the data warehouse, the results of the query are produced by combining or joining one or more dimension tables with the fact table.
- The joins are between the fact table and individual dimension tables.
- The relationship of a particular row in the fact table is with the rows in each dimension table.

Star Schema-example of order analysis

- Take a simple query against the STAR schema. Let us say that the marketing department wants the quantity sold and order dollars for product *bigpart-1*, relating to customers in the state of *Maine*, obtained by salesperson *Jane Doe*, during the month of *June*.
- Figure shows how this query is formulated from the STAR schema.

Understanding query from the star schema



Drill down analysis from the star schema

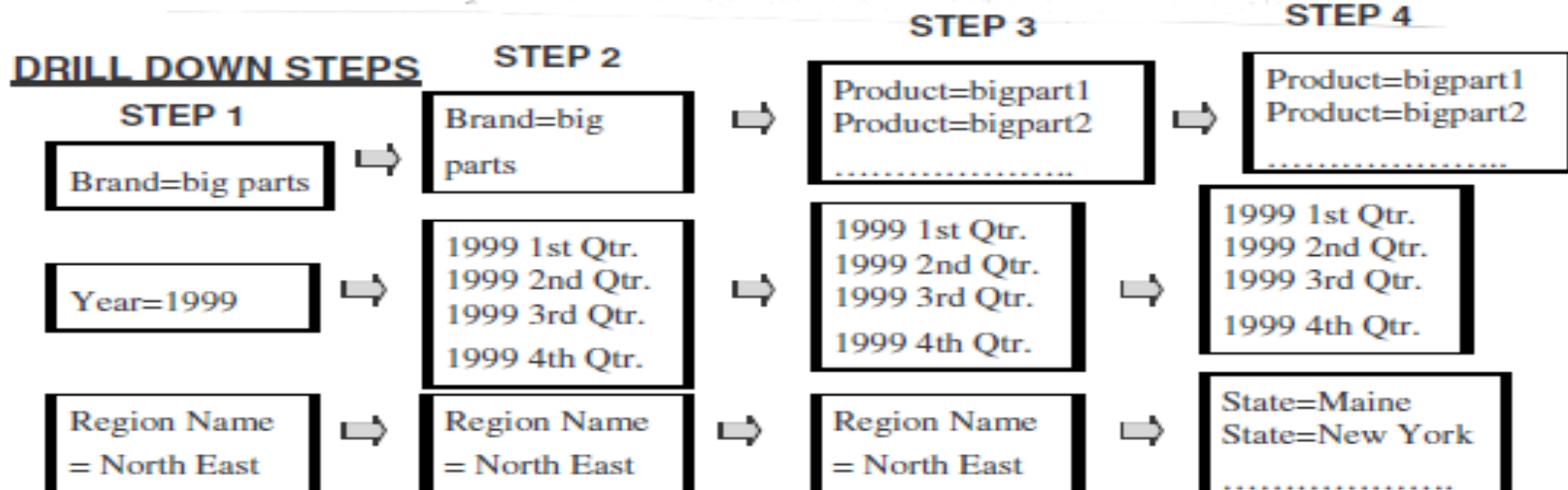
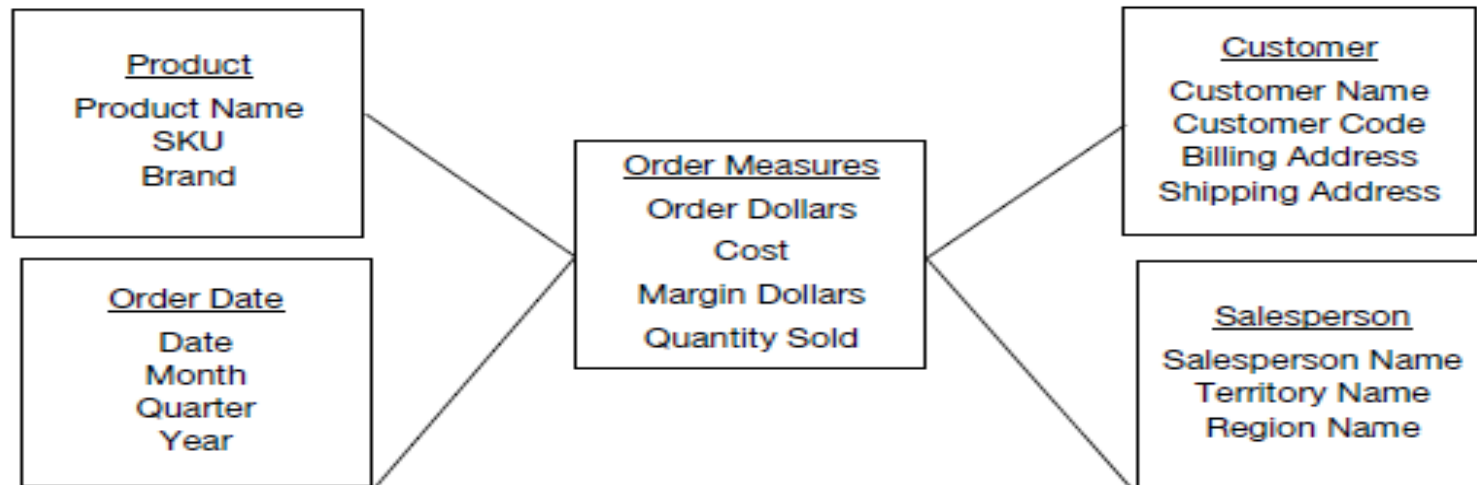
- A common type of analysis is the drilling down of summary numbers to get at the details at the lower levels.

Let us say that the marketing department has initiated a specific analysis by placing the following query:

Show me the total quantity sold of product brand *big parts* to customers in the *Northeast Region* for year 1999.

- In the next step of the analysis, the marketing department now wants to drill down to the level of *quarters in 1999* for the *Northeast Region* for the same product brand, *big parts*.
- Next, the analysis goes down to the level of individual products in that brand. Finally, the analysis goes to the level of details by individual states in the Northeast Region.

Understanding drill down analysis from the star schema



Dimension table

- Contain information about a particular dimension.
 - Dimension table key
 - Table is wide
 - Textual attributes
 - Attributes not directly related
 - Not normalized
 - Drilling down, rolling up
 - Multiple hierarchies
 - Fewer number of records

Facts

- Numeric measurements (values) that represent a specific business aspect or activity
- Stored in a fact table at the center of the star scheme
- Contains facts that are linked through their dimensions
- Can be computed or derived at run time

Fact table

- Contains primary information of the warehouse
 - Concatenated key
 - Data grain: **level of details for the measurements.**
 - Fully additive measures
 - Semi-additive measures(derived attributes)
 - Table deep, not wide
 - Sparse data:(**The fact table rows for closed dates will not have values for the measures(null)**).
 - Degenerate dimensions(attributes which are neither fact or a dimension)

Fact table

Measures :

- **Additive:** Additive facts are facts that can be summed up through all of the dimensions in the fact table.
- **Semi-Additive:** Semi-additive facts are facts that can be summed up for some of the dimensions in the fact table, but not the others.

Table deep, not wide:

- Take a very simplistic example of 3 products, 5 customers, 30 days, and 10 sales representatives represented as rows in the dimension tables. Even in this example, the number of fact table rows will be 4500, very large in comparison with the dimension table rows.

Fact table - Example

Order_Date_Key

Customer_Key

Product_Key

*Unit Sold

*Sales_Amount

*Cost_Amount

Profit_margin

*Order_Number

*Each of these relates to a particular product on a certain date for a specific customer procured.

*When you pick up attributes for the dimension tables and the fact tables from operational systems, you will be left with some data elements in the operational systems that are neither facts nor strictly dimension attributes. These attributes are useful in some types of analyses. For example, you may be looking for average number of products per order. Then you will have to relate the products to the order number to calculate the average.

Fact table-Example

Suppose that a company sells products to customers.

Every sale is a fact that happens. The purpose of this table is to record the units sold of each product to each customer on a daily basis.

Unit Sold is an additive fact, because you can sum up this fact along any of the three dimensions present in the fact table -- time, customer, and product. For example, the sum of **Unit Sold** for all 7 days in a week represents the total units sold for that week.

Order_Date_Key	Product_Key	Customer_Key	Unit Sold
4	17	2	1
8	21	3	2
8	4	1	1
3	4	2	5

Dimension table-Example

Dimension table about customers:

Customer ID	Name	Gender	Income	Education	Region
1	Brian Edge	M	2	G	4
2	Fred Smith	M	3	D	1
3	Sally Jones	F	1	G	3
4	Tom	F	6	PG	2

Dimension table-Example

Dimension table about products:

Product ID	Product Name	Brand
4	Product_A	Samsung
10	Product_B	LG
17	Product_C	Samsung
21	Product_D	Sony

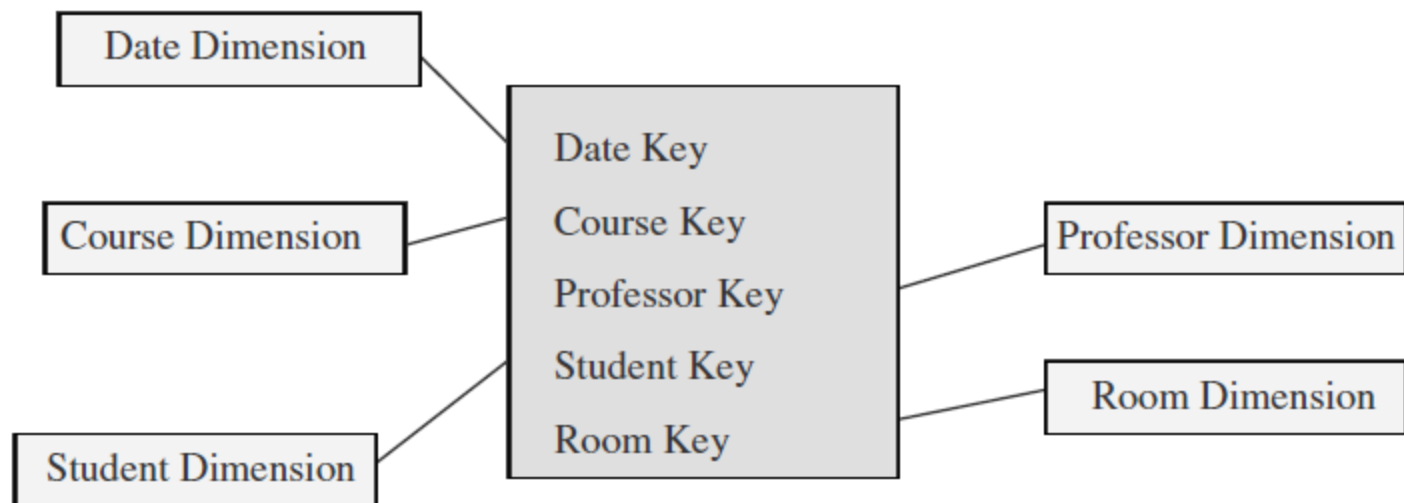
Dimension table-Example

Dimension table about time:

Time Id	Week	month	year
1	1	1	2001
2	1	1	2001
3	1	1	2001
4	1	1	2001
5	1	1	2001
6	1	1	2001
7	1	1	2001
8....30	2	1	2001

Factless fact table

- A fact table is said to be empty if it has no measures to be displayed. Fact table represents **events**
- Contains no data, only keys.



Factless fact table

- Let us say we are building a fact table to track the attendance of students. For analyzing student attendance, the possible dimensions are student, course, date, room, and professor. The attendance may be affected by any of these dimensions. When you want to mark the attendance relating to a particular course, date, room, and professor, what is the measurement you come up for recording the event? In the fact table row, the attendance will be indicated with the number *one*. Every fact table row will contain the number *one* as attendance.
- If so, why bother to record the number *one* in every fact table row? There is no need to do this. The very presence of a corresponding fact table row could indicate the attendance.
- This type of situation arises when the fact table represents events. Such fact tables really do not need to contain facts. They are “factless” fact tables.

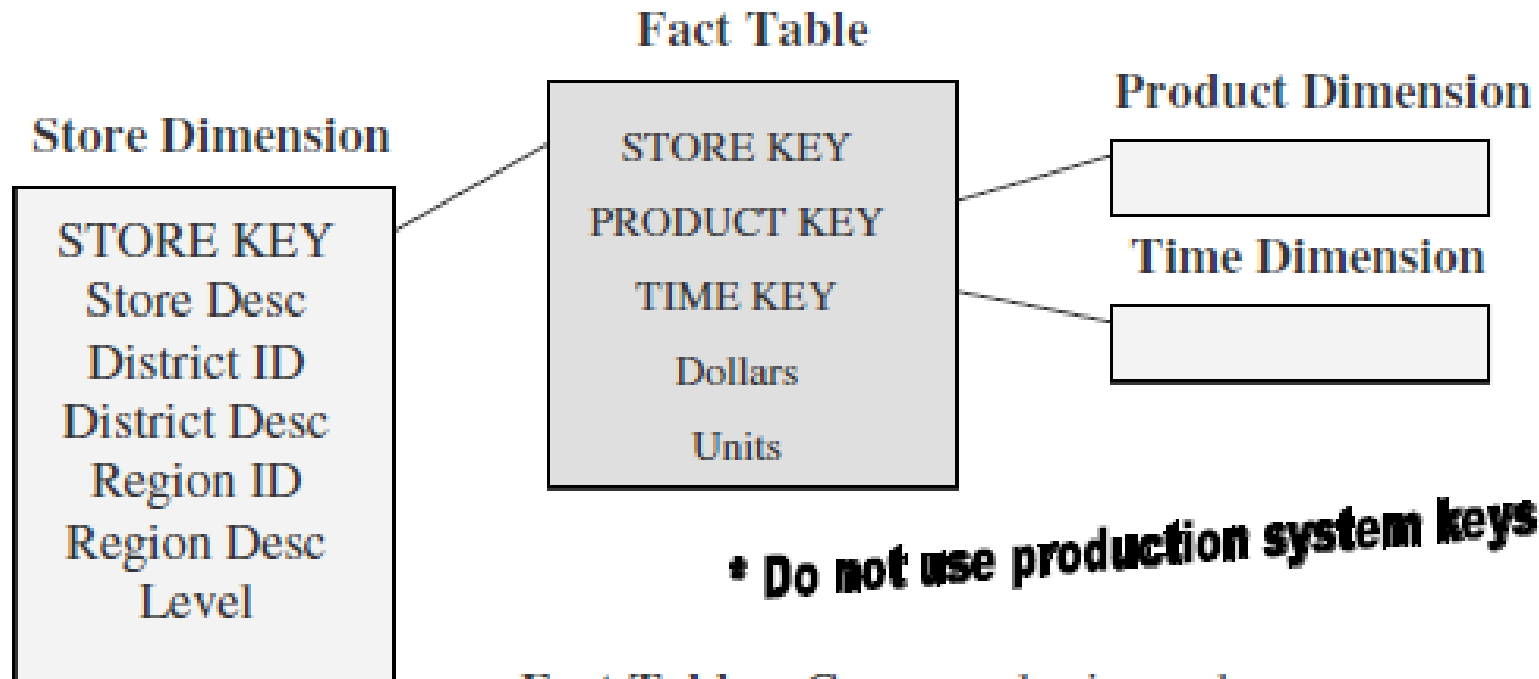
Data Granularity

- If the fact table is at the lowest grain, the users can as well drill down to the lowest grain of details.
- A single row in the fact table **should contain measurements at the lowest level for an individual product, ordered on a specific date**, relating to an individual customer, and procured by an individual sales representative.
- Let us say we want to add a new attribute of district in the sales representative dimension. This change will not warrant any changes in the fact table rows.

Advantage

- Easier to extract from operational data and load into DW.
- Compromise on the storage and maintenance of DW.

Keys in the Data Warehouse Schema



*** Do not use production system keys ***

Fact Table: Compound primary key, one segment for each dimension

Dimension Table: Generated primary key

Keys in the Data Warehouse Schema

- The first principle to follow is: Avoid built-in meanings in the primary key of the dimension tables.
- The second principle is: Do not use production system keys as primary keys for dimension tables.
- The general practice is to keep the operational system keys as additional attributes in the dimension tables.

Keys in the Data Warehouse Schema

- Primary keys: Each row in a dimension table is identified by a unique value of an attribute designated as the primary key of the dimension. In a product dimension table, the primary key identifies each product uniquely.
- Surrogate keys: System generated sequence numbers having no built-in meanings. The surrogate keys will be mapped to the production system keys.
- Foreign keys: Each dimension table is in a one-to-many relationship with the central fact table. So the primary key of each dimension table must be a foreign key in the fact table.

Primary key for Fact table

1. A single compound primary key *whose length is the total length of the keys of the individual dimension tables.* Under this option, in addition to the compound primary key, the separate foreign keys must also be kept in the fact table as additional attributes. This option increases the size of the fact table.

2. Concatenated primary key *that is the concatenation of all the primary keys of the dimension tables.* Here you need not keep the primary keys of the dimension tables as additional attributes to serve as foreign keys. The individual parts of the primary keys themselves will serve as the foreign keys.

3. A generated primary key independent of the keys of the dimension tables. In addition to the generated primary key, the foreign keys must also be kept in the fact table as additional attributes. This option also increases the size of the fact table.

- In practice, option (2) is used in most fact tables. This option enables you to easily relate the fact table rows with the dimension table rows.

Star Schema characteristics

- Star schema is a relational model with one-to-many relationship between the fact table and the dimension tables.
- De-normalized relational model
- Easy to understand. Reflects how users think. This makes it easy for them to query and analyse the data.
- Optimizes navigation.
- Enhances query extraction.
- Ability to drill down or roll up.

Advantages of the star schema

- Easy for users to understand
- Optimizes navigation
- Most suitable for query processing

Dimensional modeling: Advanced Topics

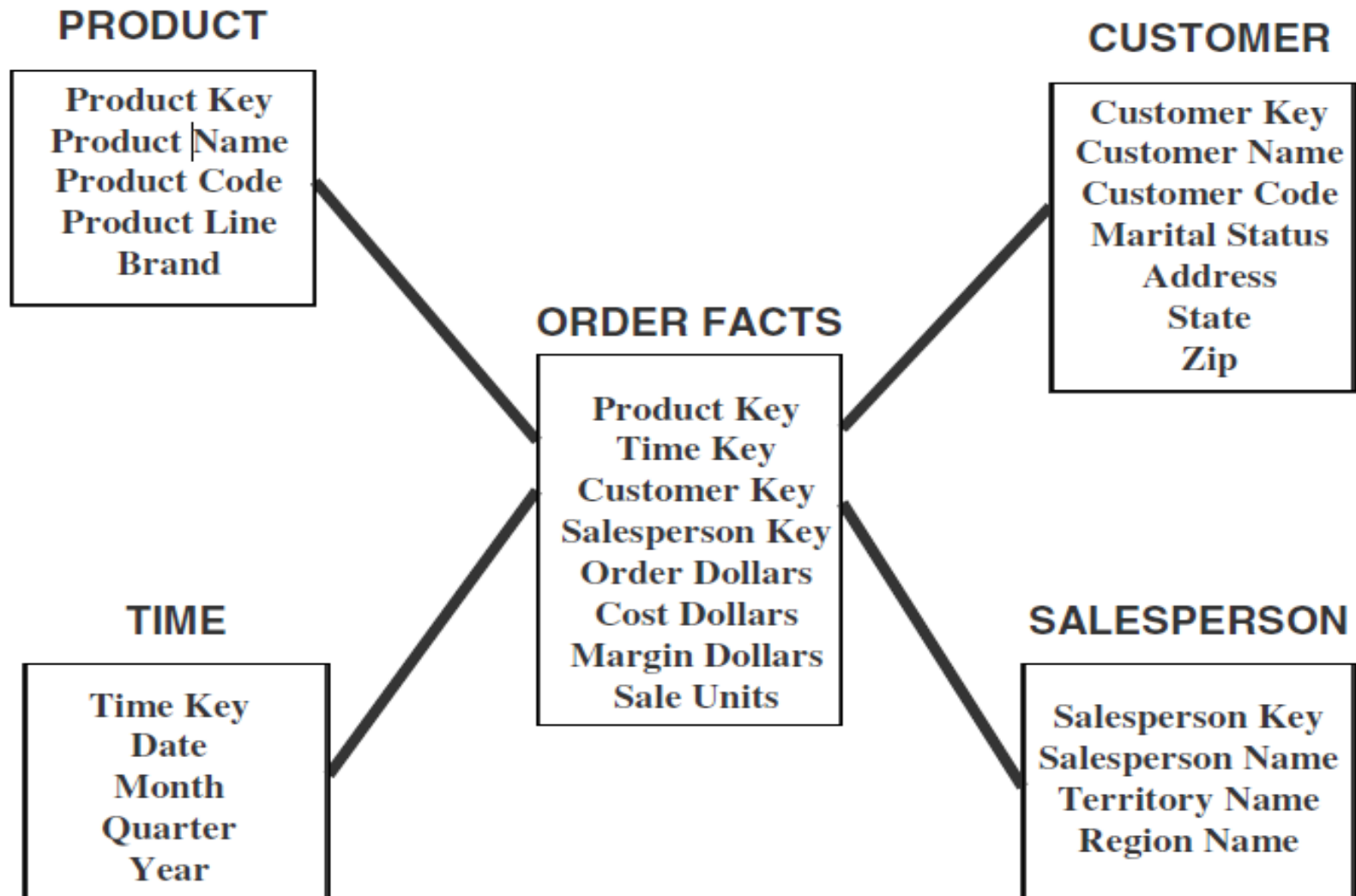


Figure 11-1 STAR Schema for order tracking.

Objectives

- Slowly changing dimensions
- Large dimensions
- Snowflake schema
- Aggregate tables
- Family of stars and their applications

Updating the Dimension table

- Every day as more and more sales take place, more and more rows get added to the fact table. The fact table continues to grow in the number of rows over time.
- Compared to the fact table, the dimension tables are more stable and less volatile.
- However, unlike the fact table, which changes through the increase in the number of rows, a dimension table does not change just through the increase in the number of rows, but also through changes to the attributes themselves.

Updating the Dimension table

- More rows are added to the Dimension tables over time.
- Changes to certain attributes of a row become important at times.
- There are many types of changes that affect the dimension tables.

Changing Dimensions- Examples

- The **product category** for a product was changed.
- Consider the customer dimension table. What happens **when a customer's status changes from rental home to own home**? The corresponding row in that dimension table must be changed.
- The payment method dimension table. When **finance type changes for one of the payment methods**, this change must be reflected in the payment method dimension table.

Slowly changing dimensions: Principles

- Most dimensions are generally constant over time
- Many dimensions, though not constant over time, change slowly
- The product key of the source record does not change
- The description and other attributes change slowly over time
- In the source OLTP systems, the new values overwrite the old ones
- Overwriting of dimension table attributes is not always the appropriate option in a data warehouse
- The ways changes are made to the dimension tables depend on the types of changes and what information must be preserved in the data warehouse

Type 1: Correction of errors

The general principles for Type 1 changes:

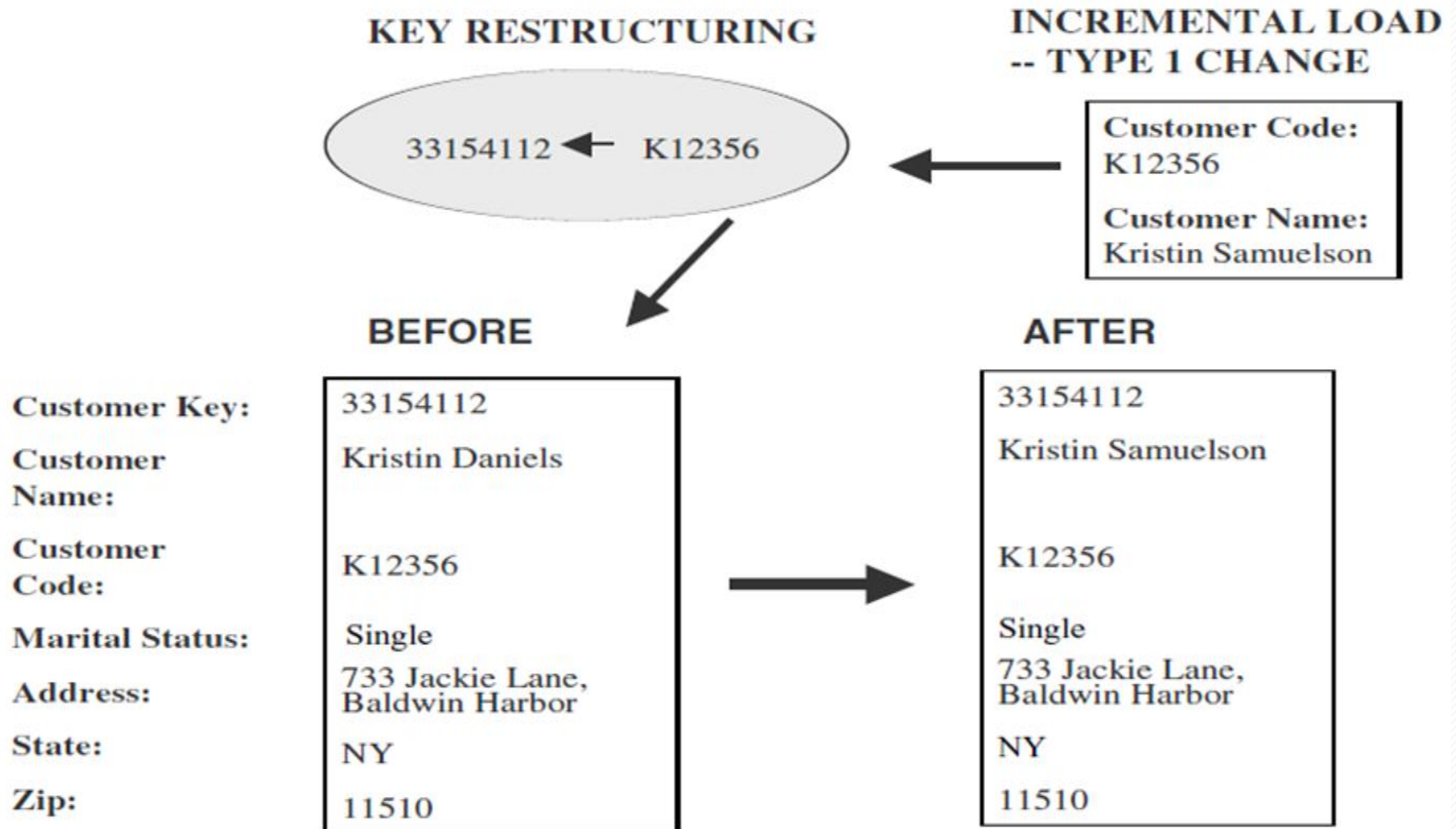
- Usually change relate to correction of errors in the source systems.
- E.g., spelling error in customer names; change of names of customers;
- There is no need to preserve the old values here.
- The old value in the source system needs to be discarded.
- **The changes made need not be preserved or noted.**

Type 1: Correction of errors

The method for applying Type 1 changes is:

- Overwrite the attribute value in the dimension table row with the new value
- The old value of the attribute is not preserved
- No other changes are made in the dimension table row
- The **key of this dimension table or any other key values are not affected**
- This type is easiest to implement

The method for applying Type 1 changes



Type 2: Preservation of History

The general principles for this type of change:

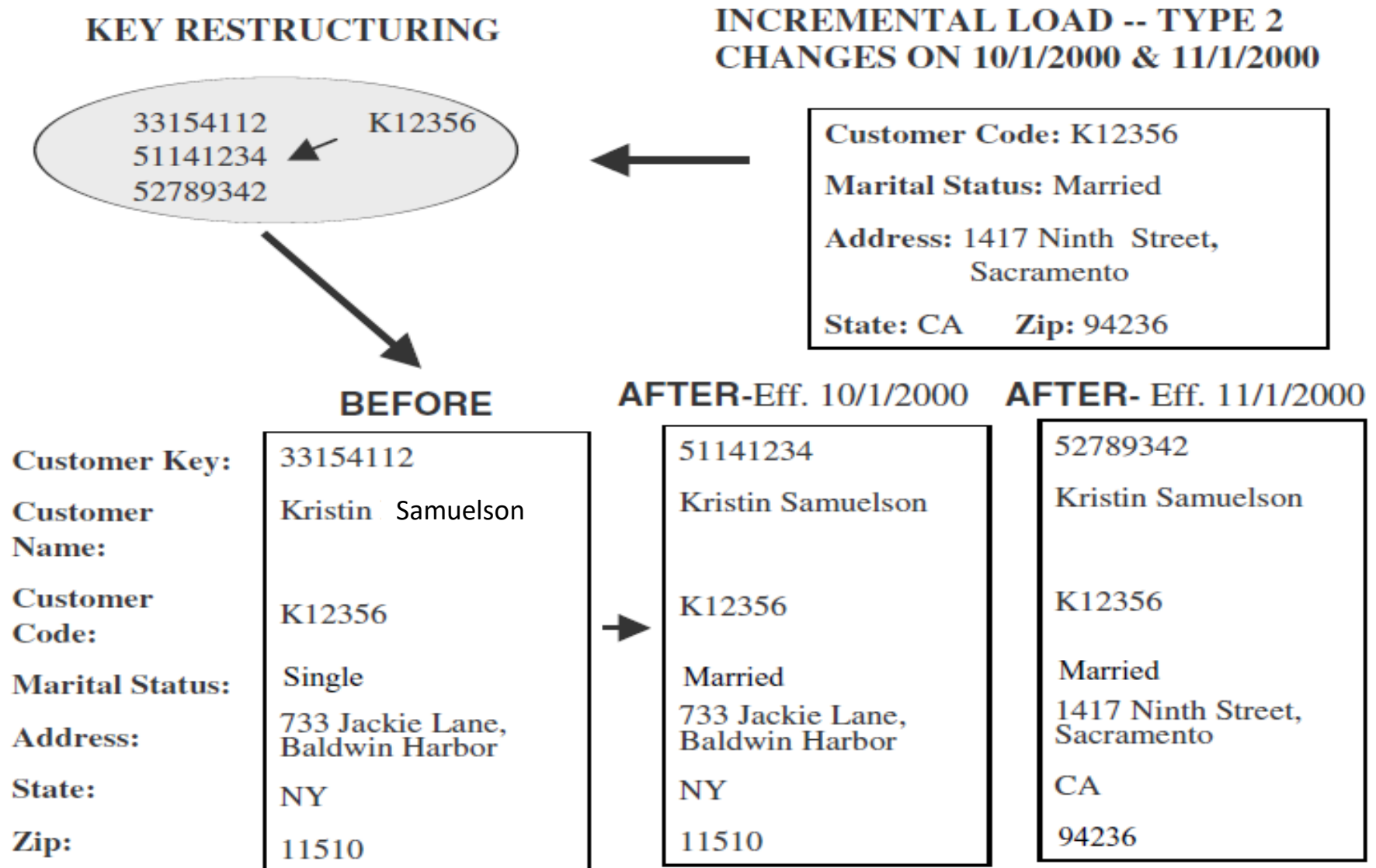
- True changes in the source systems.
- E.g., change of marital status; change of address
- There is a need to preserve history
- This type of change partitions the history in the data warehouse
- Every change for the same attribute must be preserved.

The method for applying Type 2 changes

The method for applying Type 2 changes is:

- Add a new dimension table row with the new value of the changed attribute
- An effective date field may be included in the dimension table
- There are no changes to the original row in the dimension table
- The key of the original row is not affected
- The new row is inserted with a new surrogate key

The method for applying Type 2 changes



Type 3: Tentative Soft Revision

The general principles for Type 3 changes:

Sometimes, though rarely, there is a need to track both the old and new values of changed attributes for a certain period, in both forward and backward directions. These types of changes are Type 3 changes .

- Tentative changes in the source system
- E.g., if an employee will get posted for a short period to a different location
- There is a need to keep track of history with old and new values of the changed attribute
- Used to compare performances across the transition

Type 3 change : Example

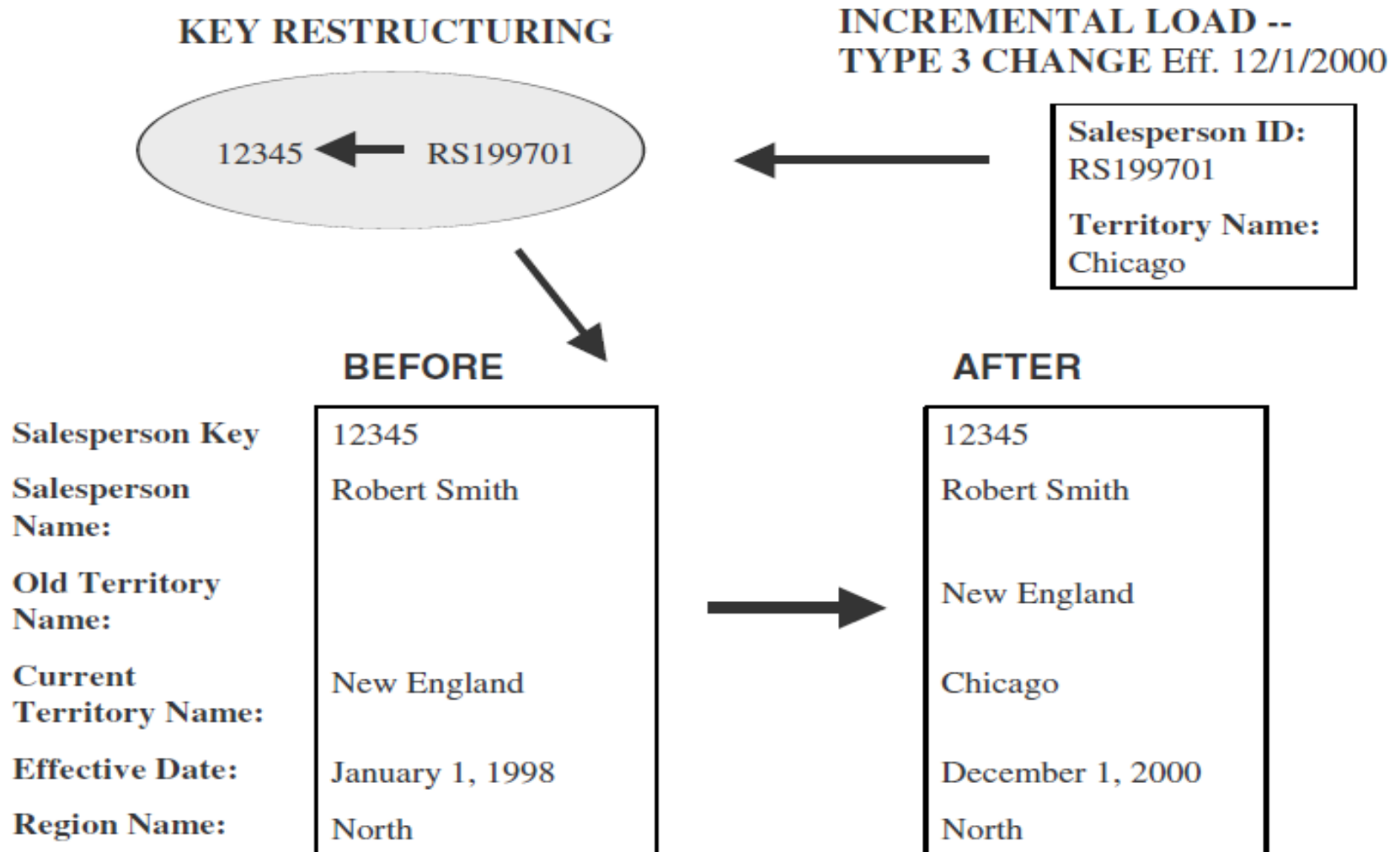
Assume your marketing department is planning a realignment of the territorial assignments for salespersons. Before making a permanent realignment, they want to count the orders in two ways: according to the current territorial alignment and also according to the proposed realignment. This type of provisional or tentative change is a Type 3 change.

The method for applying Type 3 changes

The method for applying Type 3 changes

- Add an “old” field in the dimension table for the affected attribute
- Push down the existing value of the attribute from the “current” field to the “old” field
- Keep the new value of the attribute in the “current” field
- Also, you may add a “current” effective date field for the attribute
- The key of the row is not affected

The method for applying Type 3 changes





miscellaneous dimensions

- Large dimensions
- Rapidly changing dimensions
- Junk dimensions

Large dimensions

You may consider a dimension large based on two factors:

- Very deep (large number of rows)
- Very wide (large number of attributes)
- Have multiple hierarchies
- Large dimensions call for special considerations. Because of the sheer size, many data warehouse functions involving large dimensions could be slow and inefficient.
- You need to address the following issues by using effective design methods, by choosing proper indexes, and by applying other optimizing techniques.

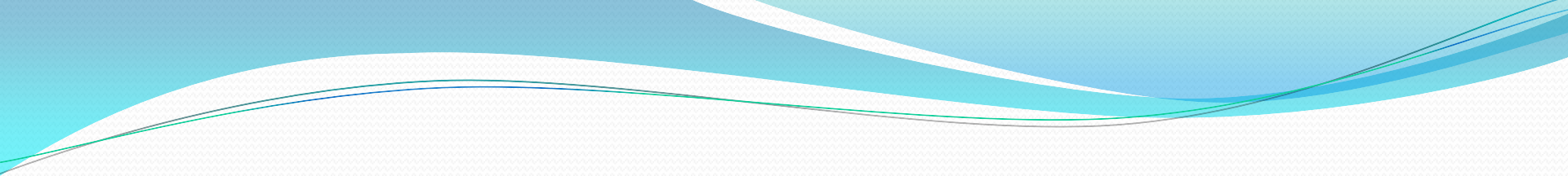
- 
- Here are some typical features of large customer and product dimensions:

Customer

- Huge—in the range of 20 million rows
- Easily up to 150 dimension attributes
- Can have multiple hierarchies

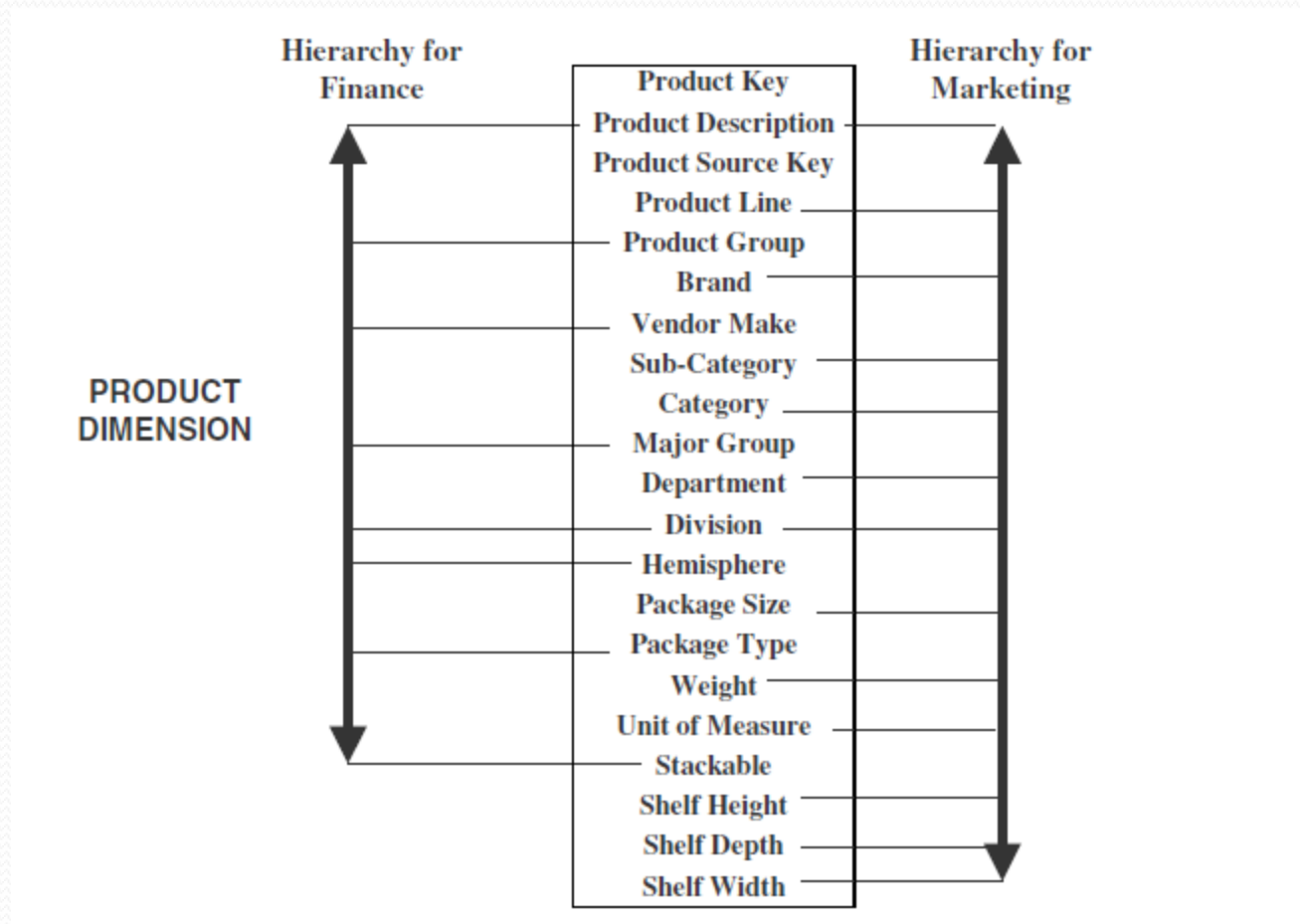
Product

- Sometimes as many as 100,000 product variations
- Can have more than 100 dimension attributes
- Can have multiple hierarchies

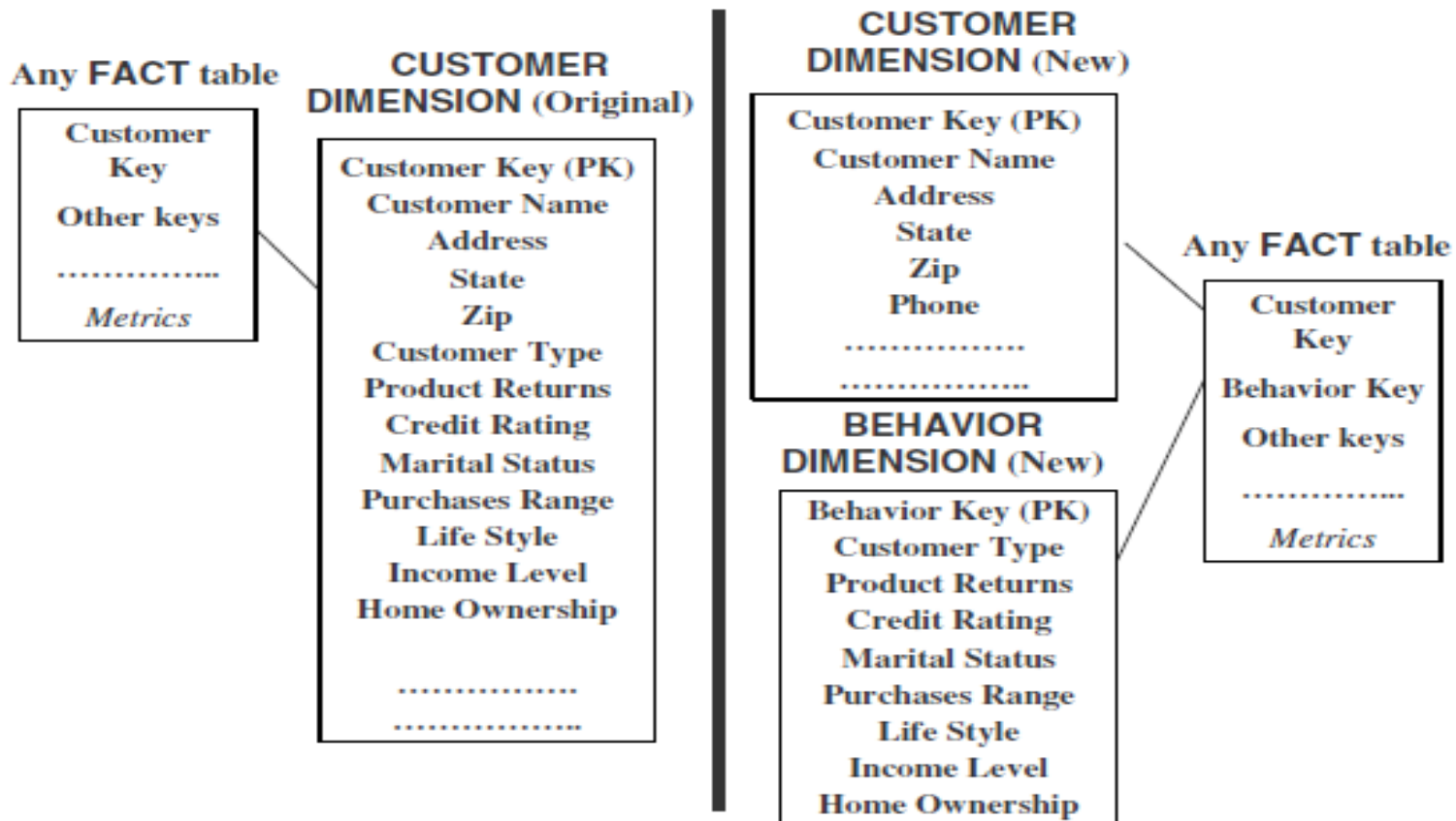


We had assumed that dimension attributes do not change too rapidly. If the change is a Type 2 change, you know that you have to create another row with the new value of the attribute. If the value of the attribute changes again, then you create another row with the newer value. What if the value changes too many times or too rapidly? Such a dimension is no longer a slowly changing dimension.

Multiple hierarchies



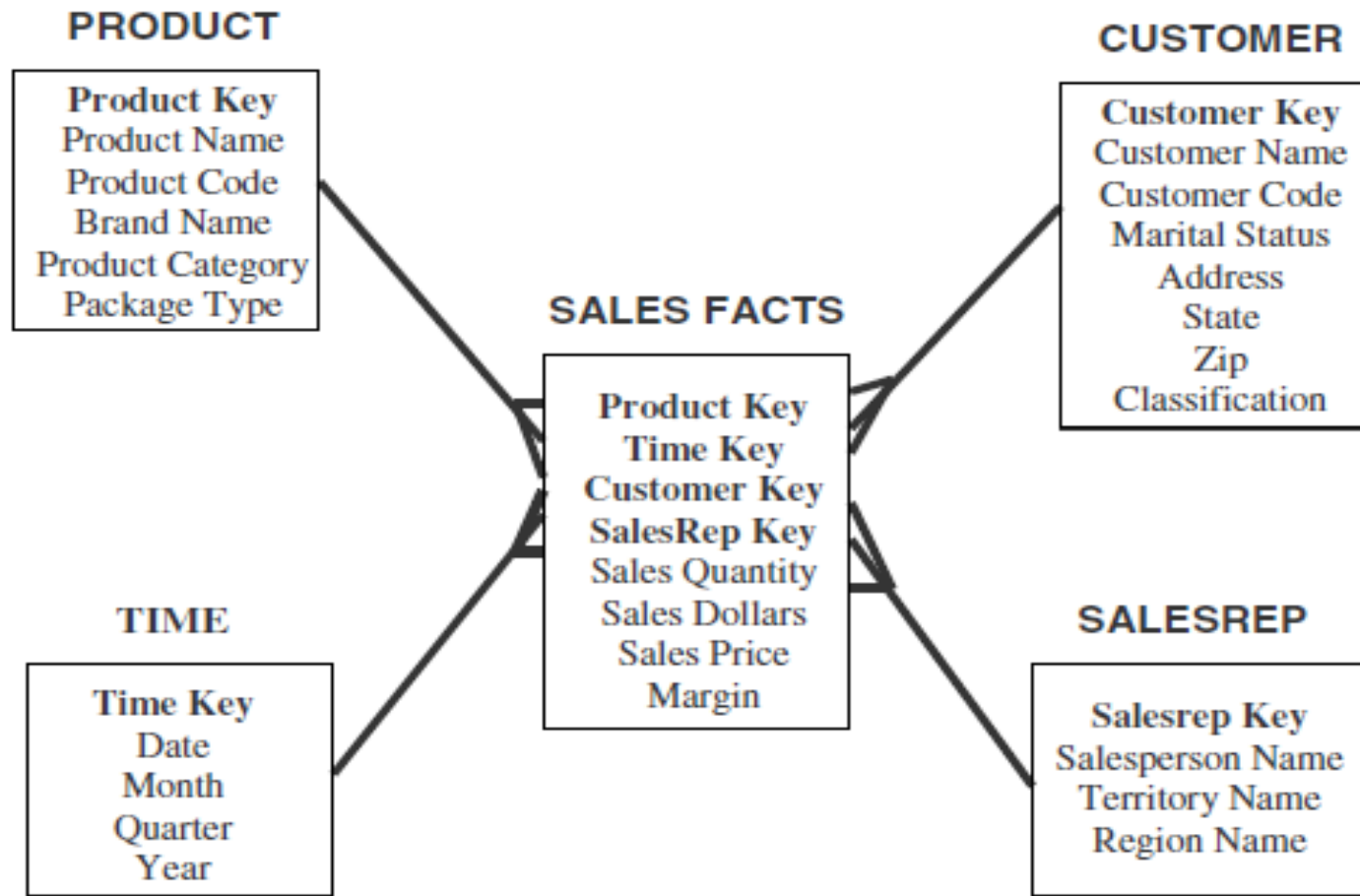
Rapidly changing dimensions



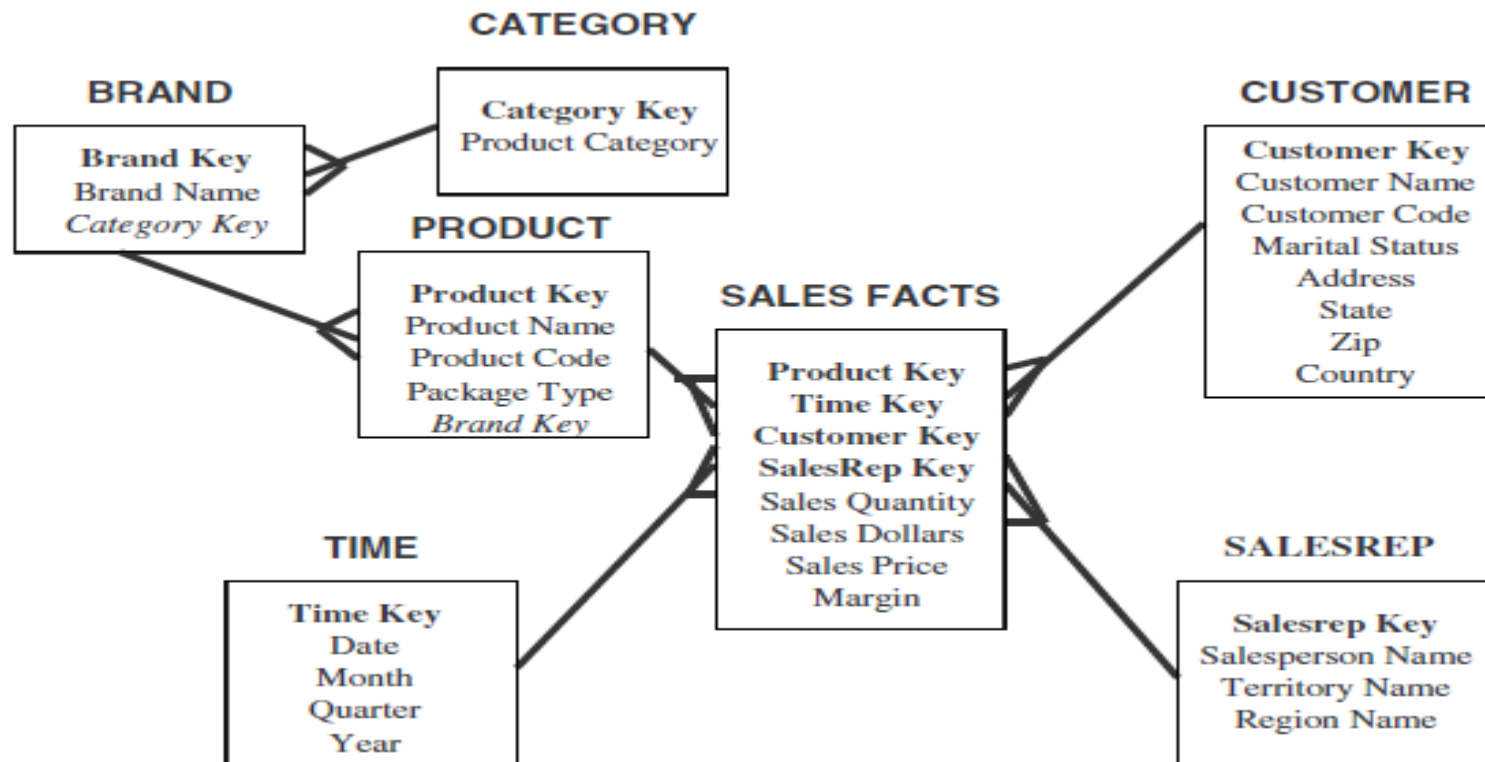
Snowflake schema

- A variation of the star schema, in which all or some of the dimension tables may be normalized.
- Eliminates redundancy
- Generally used when a dimension table is wide.
- Saves space
- Complex querying is required.

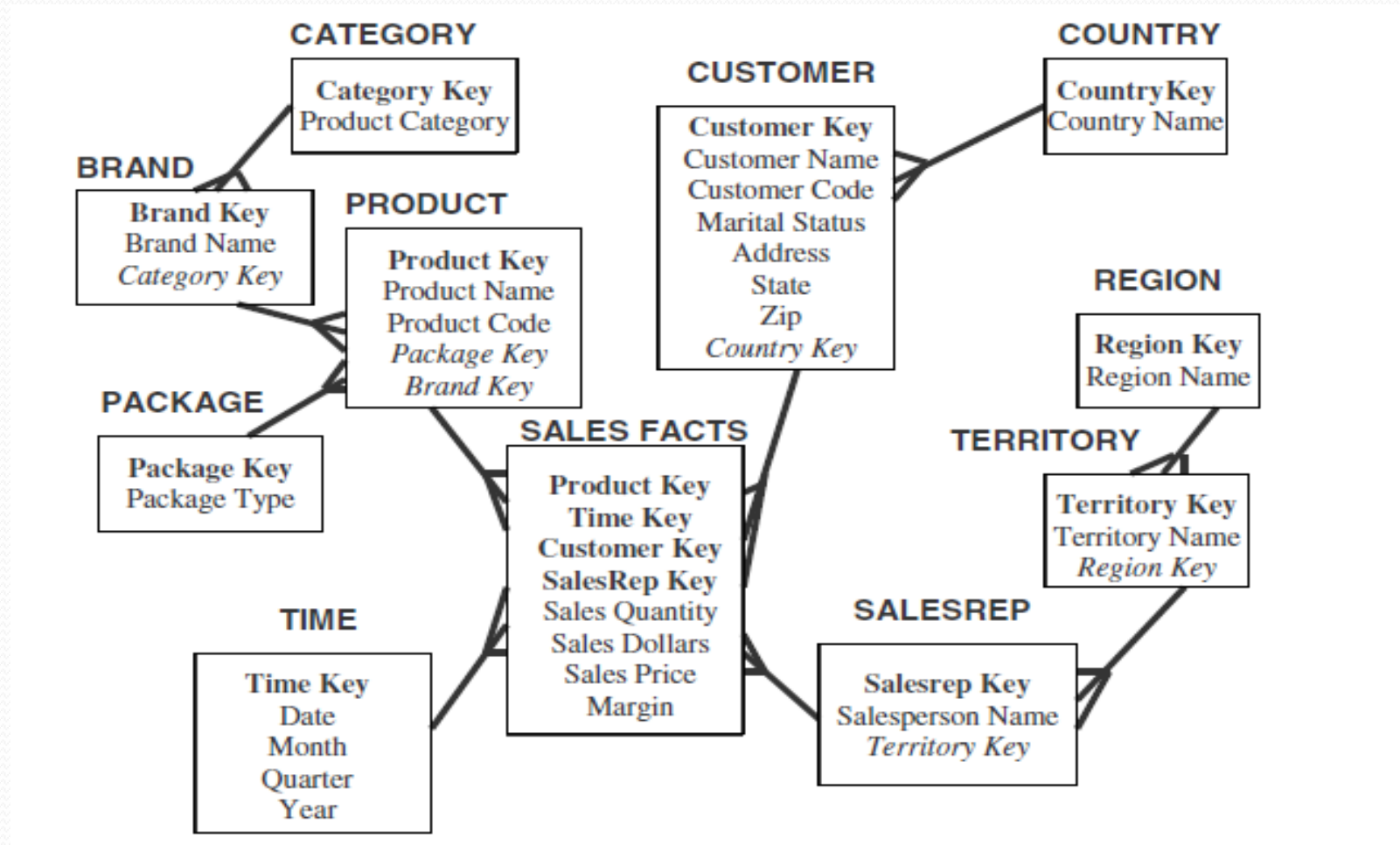
Star schema for sales



Normalized product dimension



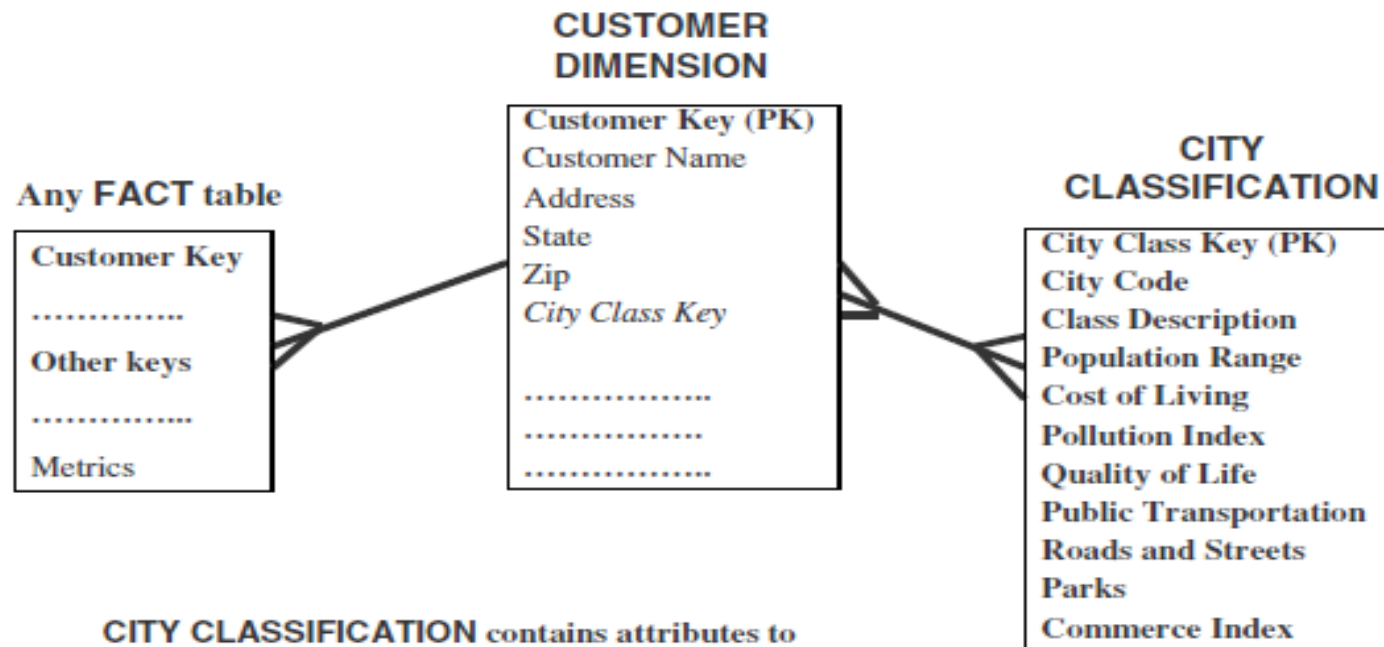
Sales snowflake schema



Advantages and disadvantages

- Advantages
 - Small savings in storage space
 - Normalized structures are easier to update and maintain
- Disadvantages
 - Schema is less sensitive
 - Browsing becomes difficult
 - Degraded query performance because of additional joins

When to snowflake



CITY CLASSIFICATION contains attributes to classify each city within a limited set of classes. These attributes are separated from the **CUSTOMER DIMENSION** to form a separate sub-dimension as **CITY CLASSIFICATION**.

Aggregate fact tables

- Contain pre-calculated summaries derived from the most granular (detailed) fact table.
- Created as a specific summarization across any number of dimensions.
- Reduces runtime processing.

Why need aggregate fact tables?

- Large size of the fact table
- To speed up query extraction
- Limitations
 - Must be re-aggregated each time there is a change in the source data
 - Do not support exploratory analysis
 - Limited interactive use.

Fact Constellation

- Multiple fact tables share dimension tables.
- This schema is viewed as collection of stars hence called galaxy schema or fact constellation.
- Sophisticated application requires such schema.

Fact Constellation (contd..)

**Sales
Fact Table**

Store Key
Product Key
Period Key
<u>Units</u>
<u>Price</u>

Product Dimension

Product Key
Product Desc

**Shipping
Fact Table**

Shipper Key
Store Key
Product Key
Period Key
<u>Units</u>
<u>Price</u>

Store Dimension

Store Key
Store Name
City
State
Region

