Datawarehouse Design: Principles of Dimensional Modeling

Building an operational system for order processing

- For gathering requirements, you interview the users in the Order Processing department .
- The users will list all the functions that need to be performed.
- They will inform you how they receive the orders, check stock, verify customers' credit arrangements, price the order, determine the shipping arrangements, and route the order to the appropriate warehouse.
- They will show you how they would like the various data elements to be presented on the GUI screen for the application. The users will also give you a list of reports they would need from the order processing application.

- In striking contrast, for a data warehousing system, the users are generally unable to define their requirements clearly. They cannot define precisely what information they really want from the data warehouse, nor can they express how they would like to use the information or process it.
- Even though the users cannot fully describe what they want in a data warehouse, they can provide you with very important insights into how they think about the business.
- They can tell you what measurement units are important for them. Each user department can let you know how they measure succes in that particular department.

Managers think in business dimensions

Marketing Vice President

How much did my new product generate
month by month, in the southern division, by user demographic, by sales office, relative to the previous version, and compared to plan?

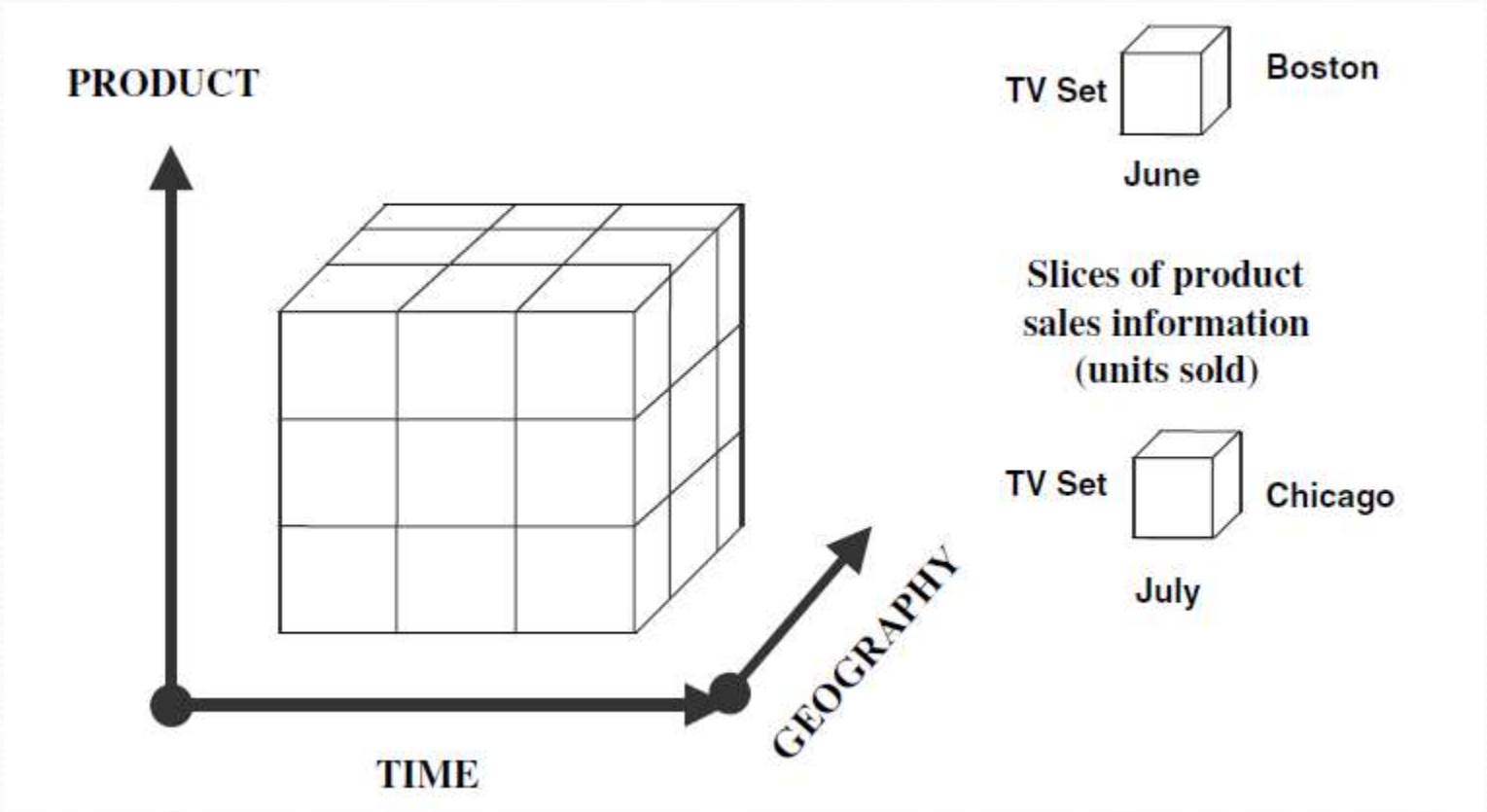
Marketing Manager

Give me sales statistics
by products, summarized by product categories, daily, weekly, and monthly, by sale districts, by distribution channels.

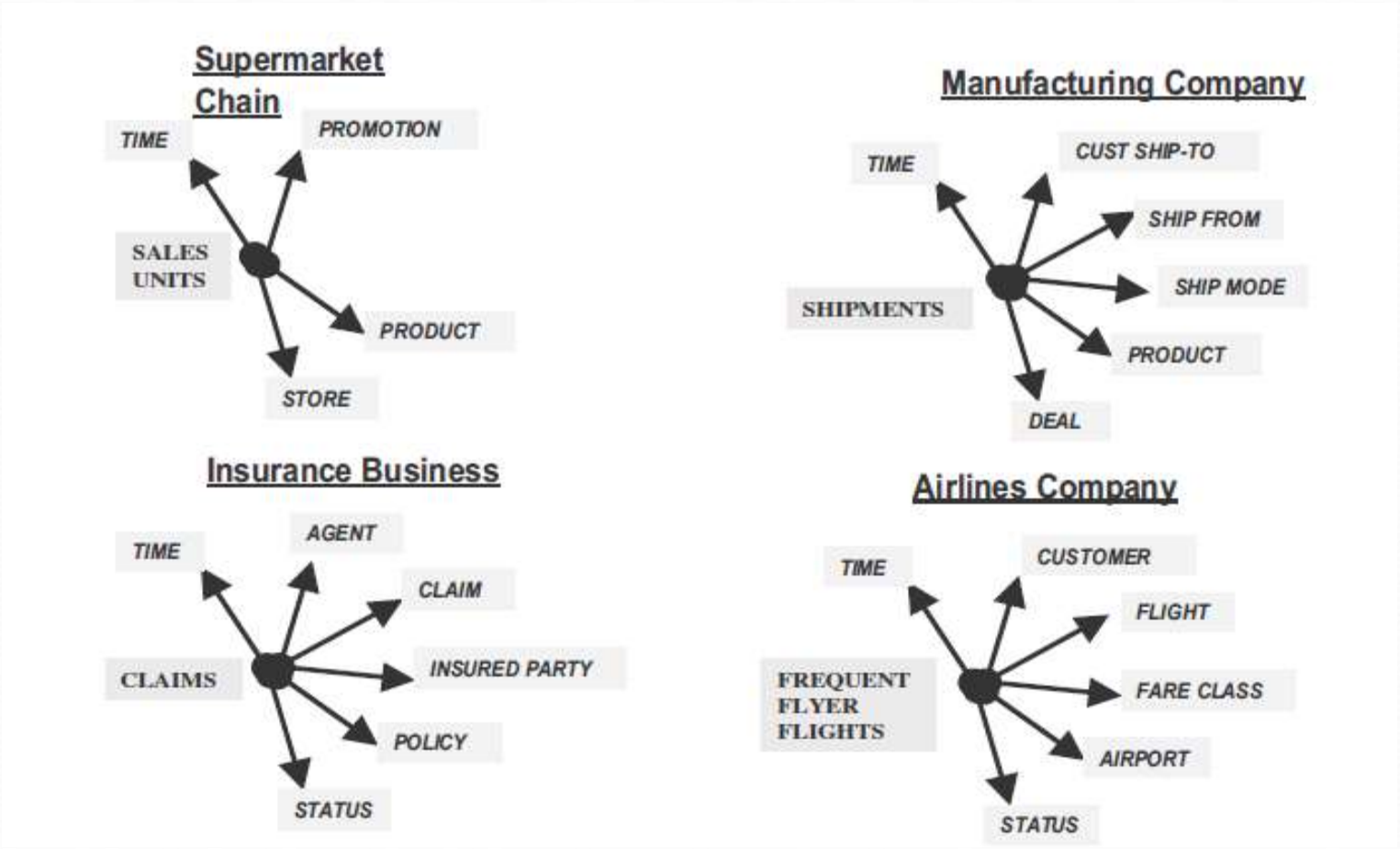
Financial Controller

Show me expenses
listing actual vs budget, by months, quarters, and annual, by budget line items, by district, division, summarized for the whole company.

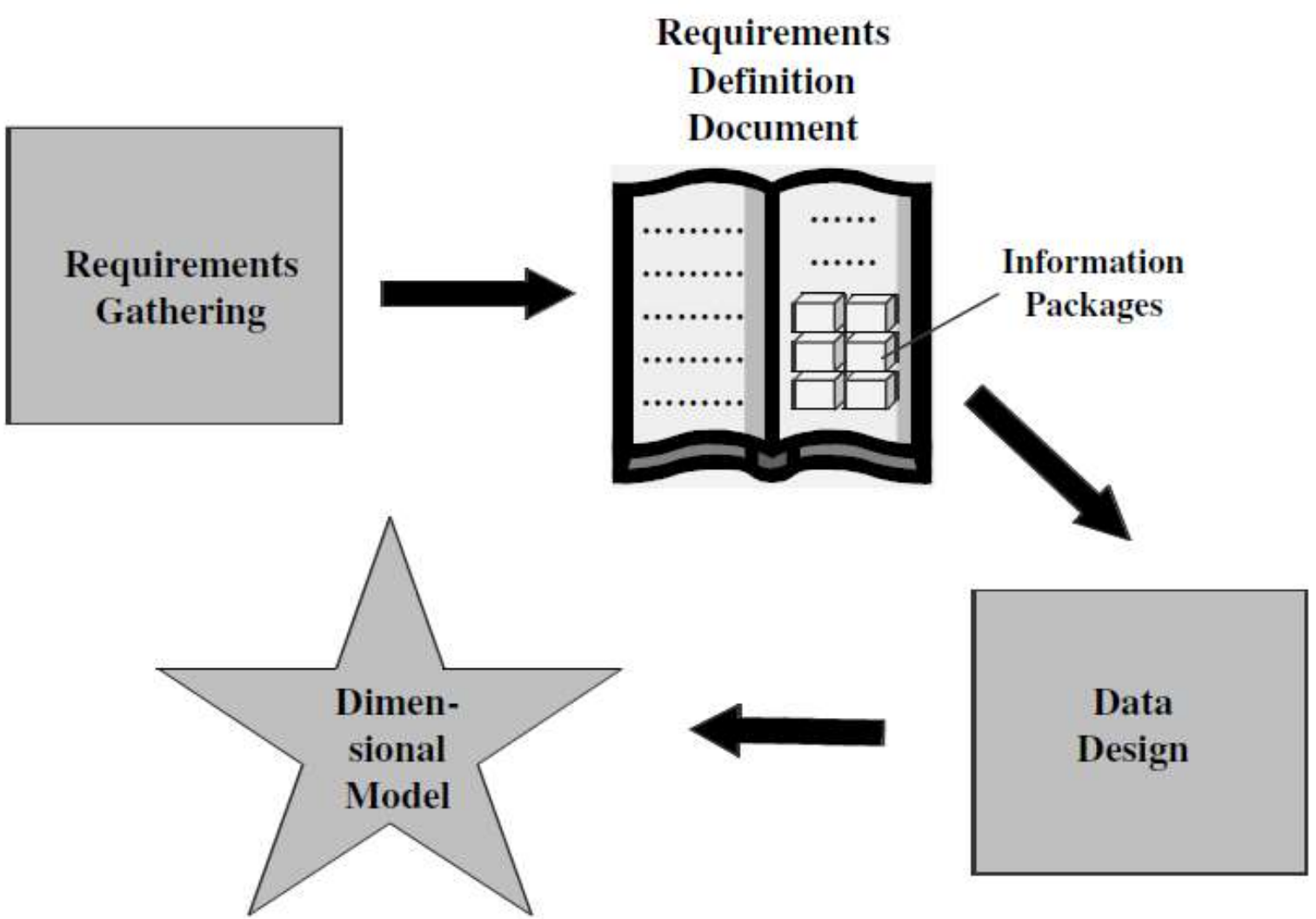
Dimensional nature of business data



Examples of business dimensions



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

Information Package Diagram

- First step is to prepare an information package, that allows the data warehouse's designers to layout the requirements for the dimension tables, their hierarchies, and the facts to be modeled.
  * AllElectronics may create a sales data warehouse in order to keep records of the store's sales with respect to the dimensions time , product , customer and store.

*Information Package :Sales

Information Subject: Sales

Dimensions

Hierarchies/Categories

Time	Product	Customer	Store
Year	Category	License type	State
Quarter	Subcategory	Category	Region
Season	Product name	Size	City
Month		Customer Name	Square footage
Date			Store name
Day of Month			
Day of Week			

Facts: Sales quantity, Item dollar amount, Item cost

Information package: automaker sales

Information Subject: Automaker Sales

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		House-hold 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

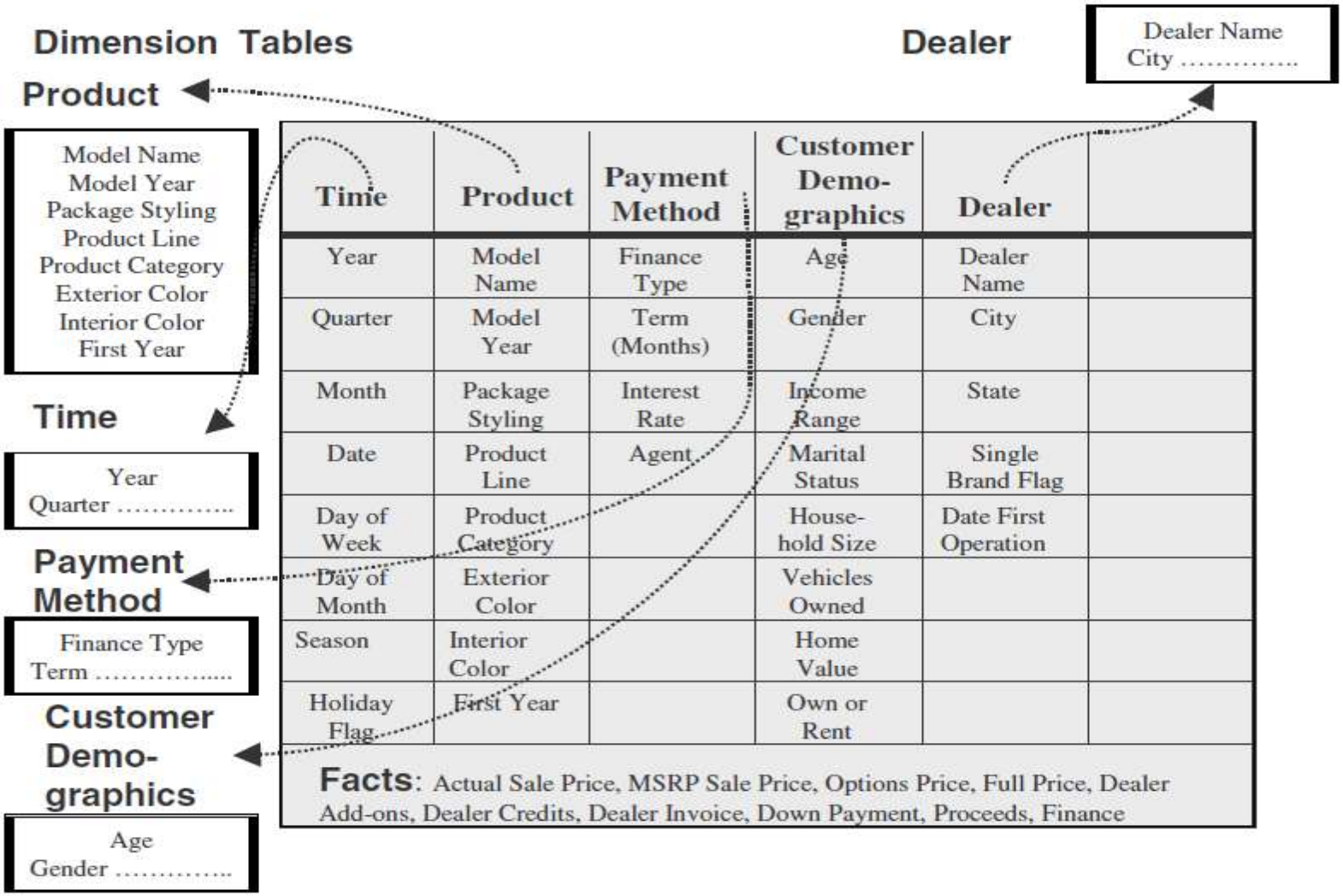
Actual Sale Price
MSRP Sale Price
Options Price
Full Price
Dealer Add-ons
Dealer Credits
Dealer Invoice
Down Payment
Proceeds
Finance

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		House-hold 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



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

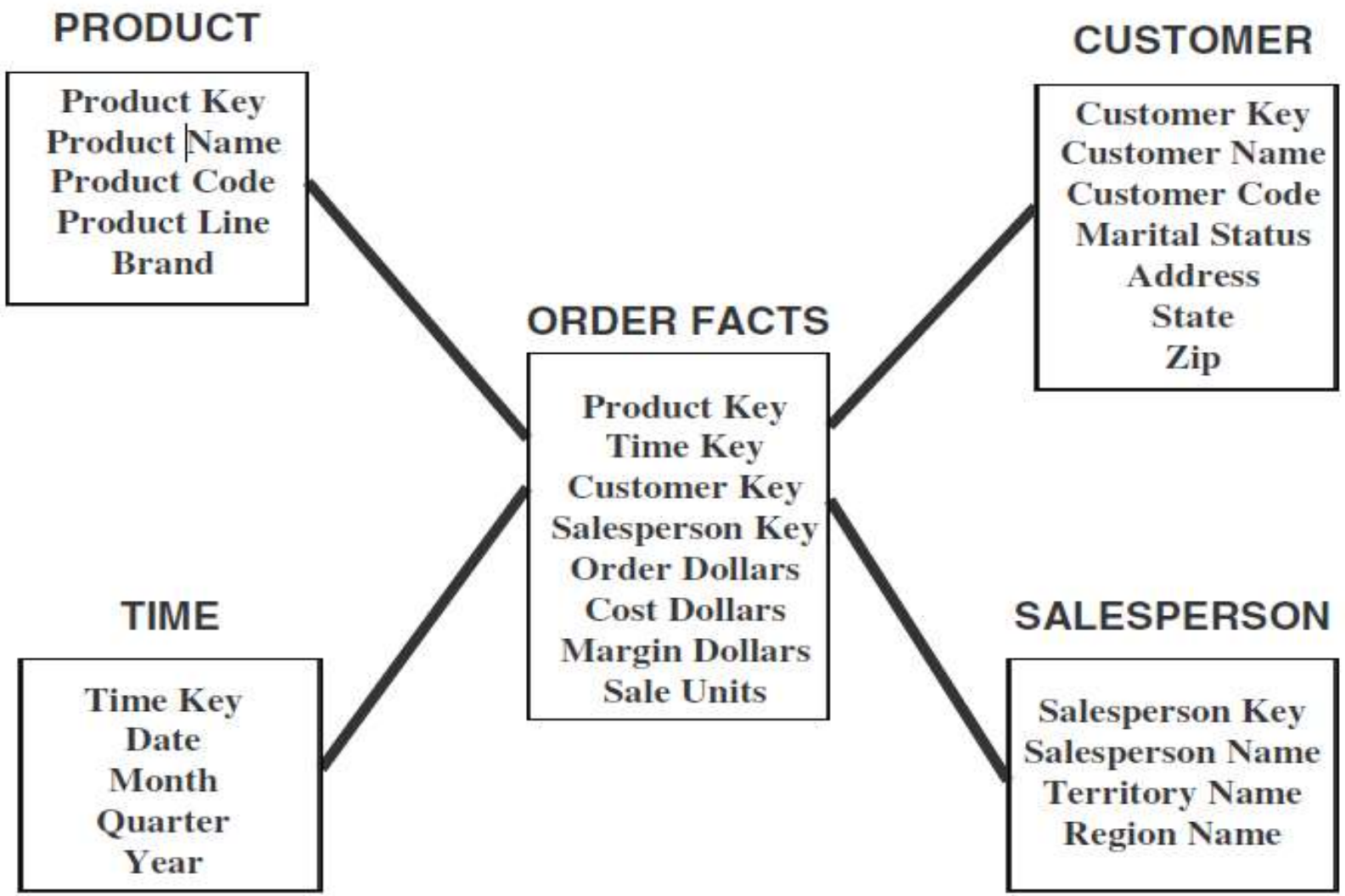
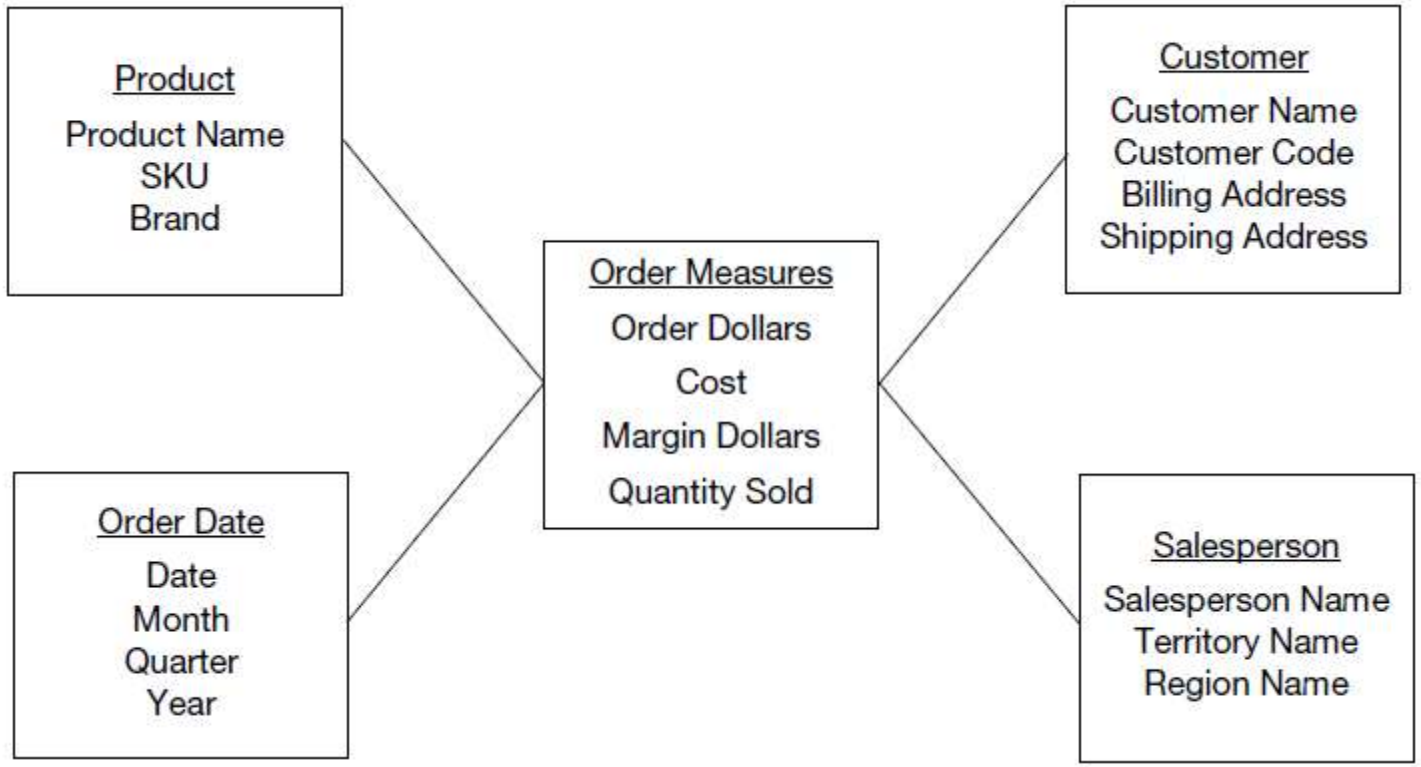


Figure 11-1 STAR Schema for order tracking.

Star Schema-example of order analysis

Query result



Star Schema-example of order analysis

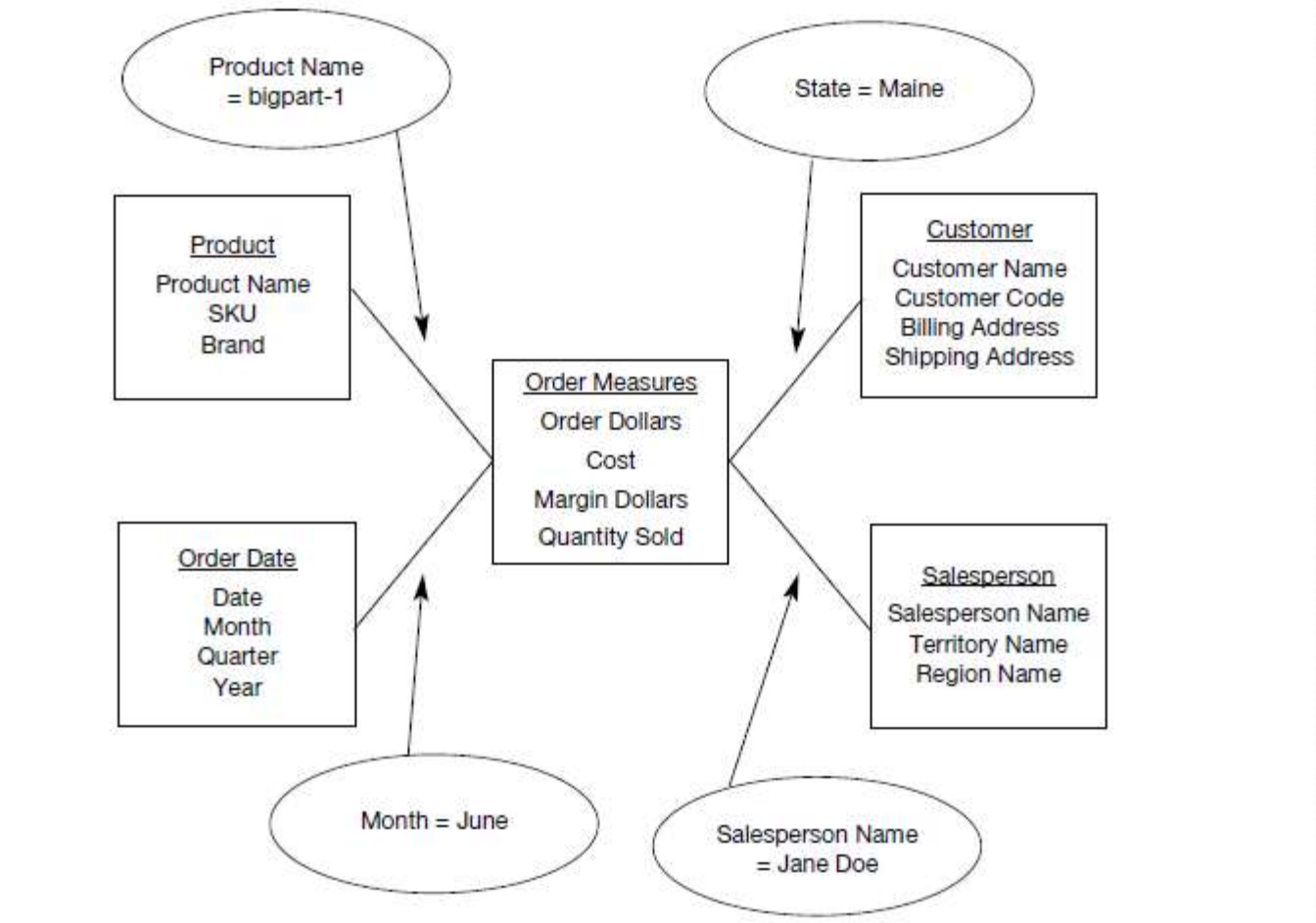
- For a given amount of dollars, what was the product sold? Who was the customer? Which salesperson brought the order? When was the order placed?
- When a query is made against the data warehouse, the results of the query are produced by combining or joining one of more dimension tables with the fact 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.

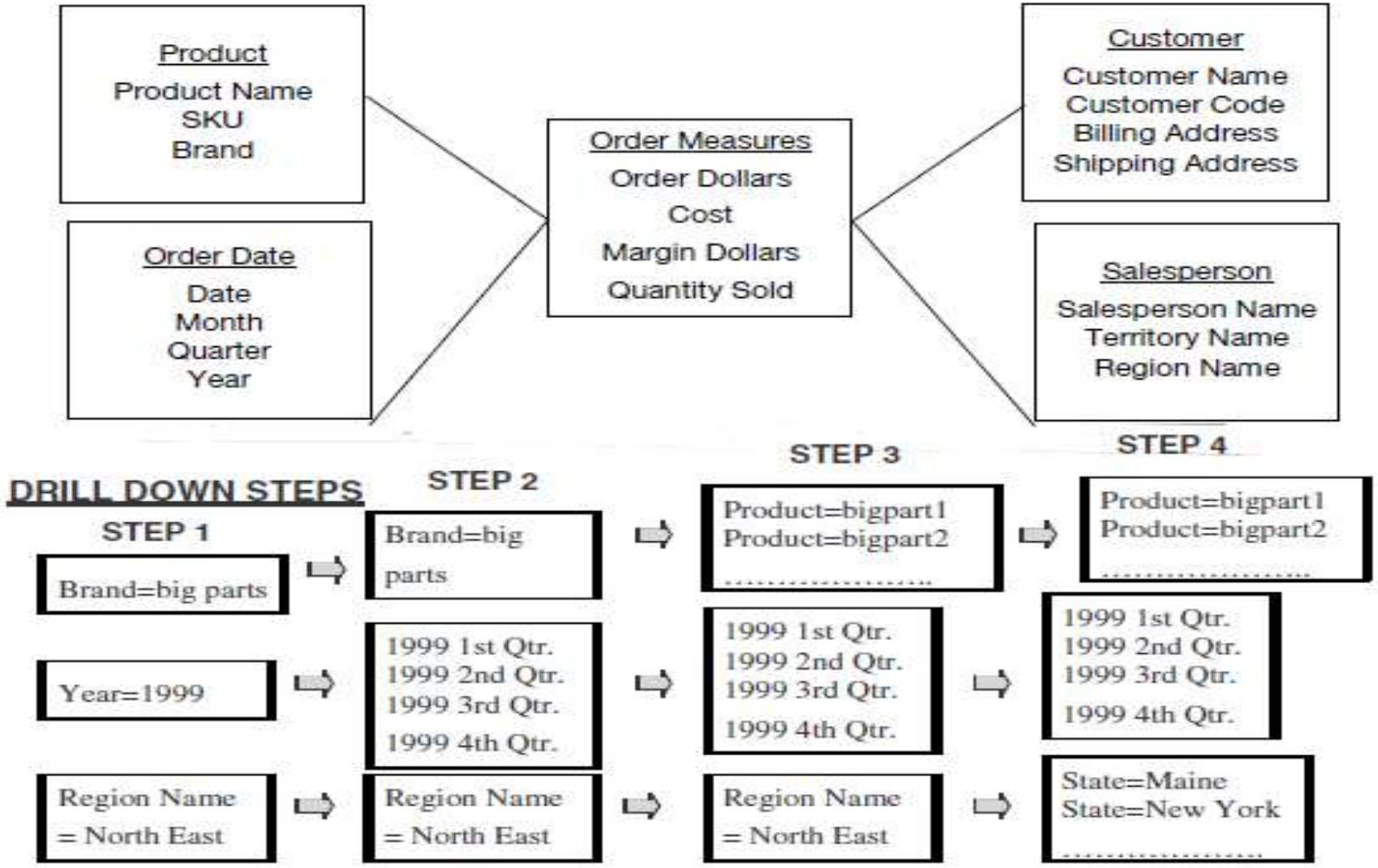
Understanding query from the star schema



Drill down analysis from the star schema

- 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(package size is not directly related to product brand)
  - Not normalized
  - Drilling down, rolling up
  - Multiple hierarchies (The product dimension table will have the attributes of marketing–product–category, marketing–product–department, finance–product–category, and finance–product–department.
  - 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

*Fully additive
* **Semi-Additive**

***Degenerate dimensions** : 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

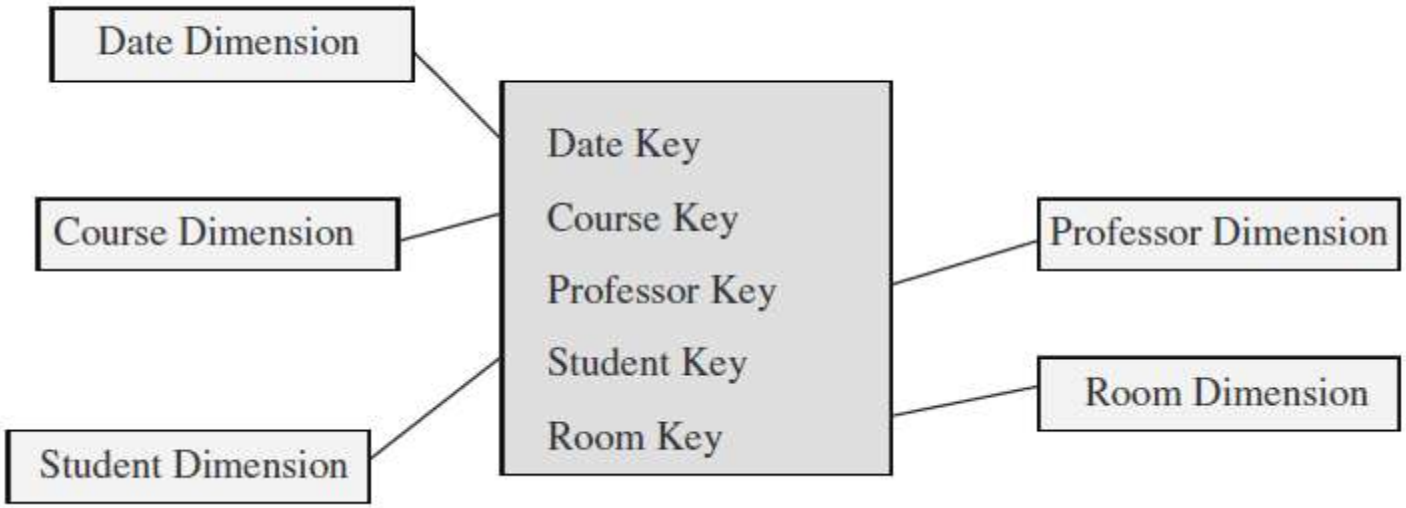
Dimension table-Example

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

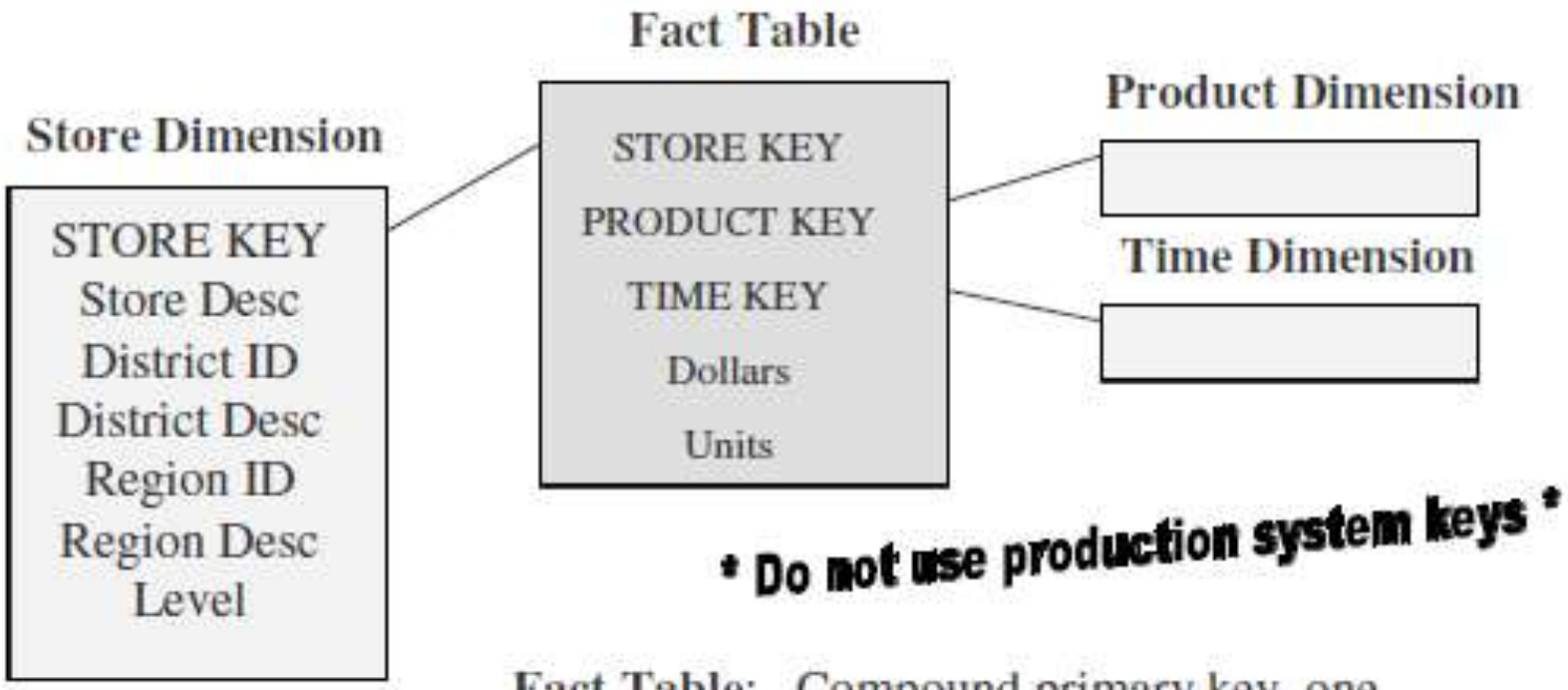
- Let us say we are building a fact table to track the attendance of students.
- For analyzing student attendance, the possible dimensions are student, course, date, room, and professor. The attendance may be affected by any of these dimensions.
- When you want to mark the attendance relating to a particular 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

- When fact table 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.

  - 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 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 analyze the data.
- Optimizes navigation.
- Enhances query extraction.
- Ability to drill down or roll up.

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:

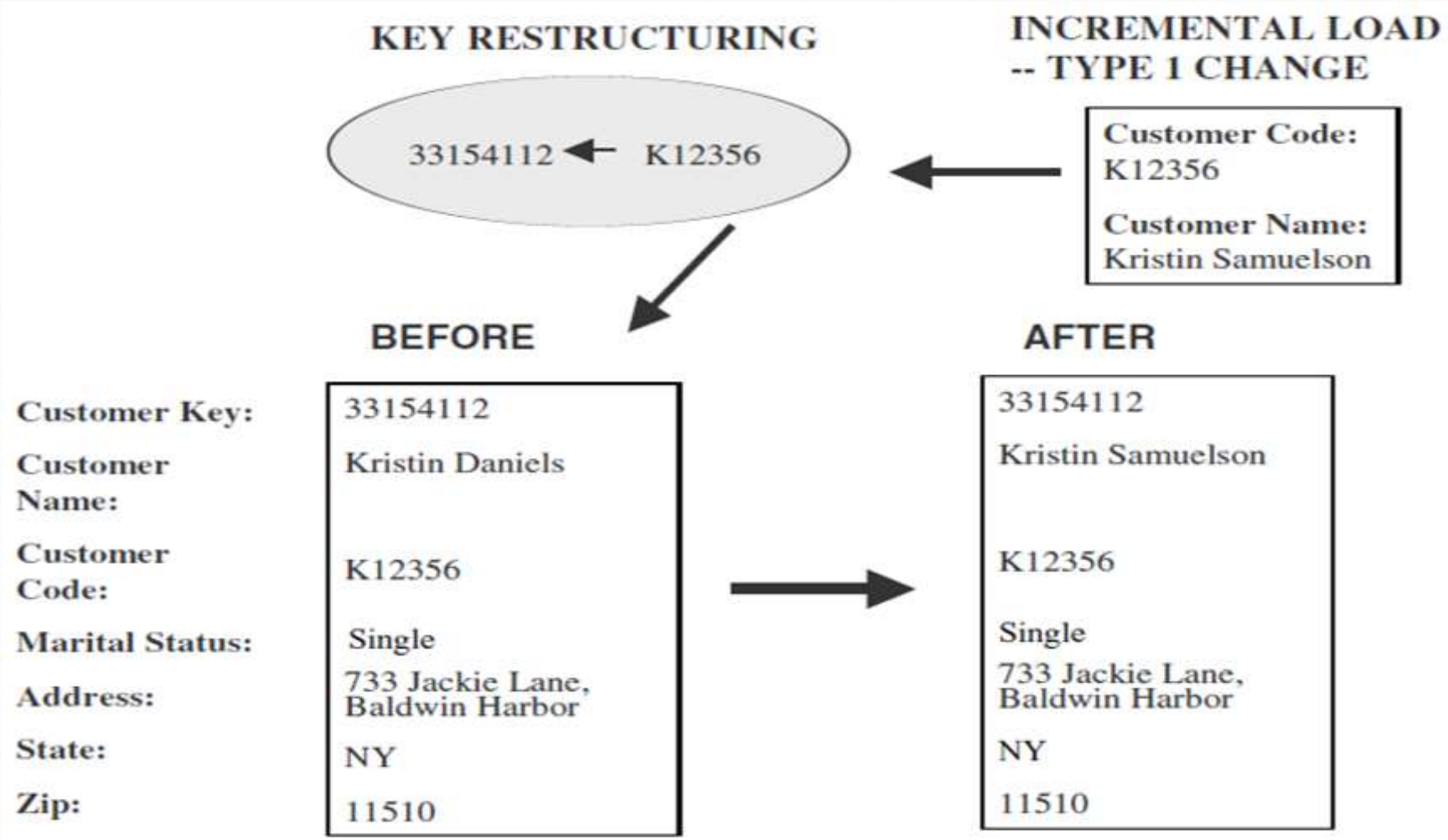
- Usually change relate to correction of errors in the source systems.
- E.g., spelling error in customer names; change of names of customers;
- There is no need to preserve the old values here.
- The old value in the source system needs to be discarded.
- The changes made need not be preserved or noted.

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:

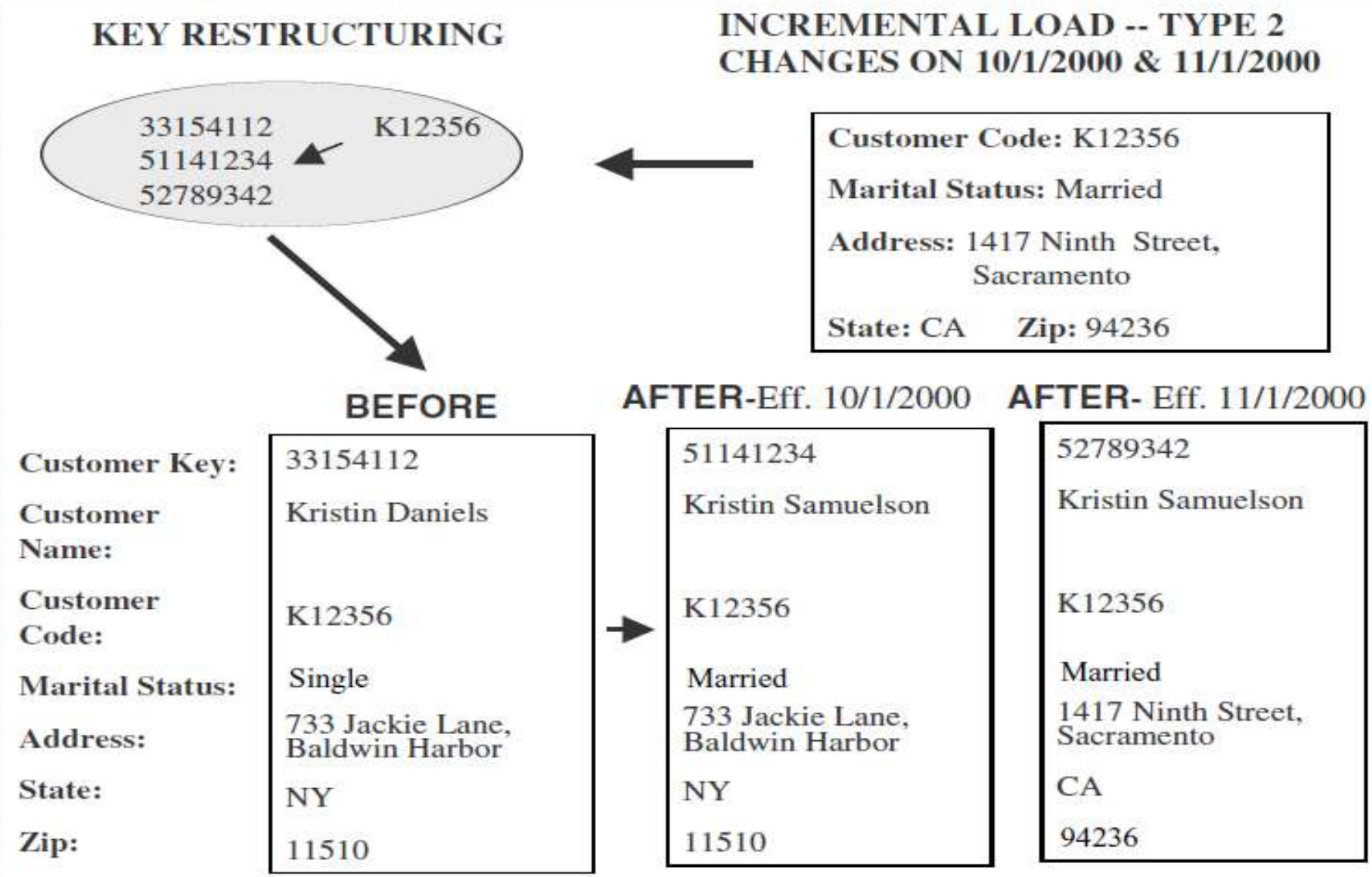
- True changes in the source systems.
- E.g., change of marital status; change of address
- There is a need to preserve history
- This type of change partitions the history in the data warehouse
- Every change for the same attribute must be preserved.

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

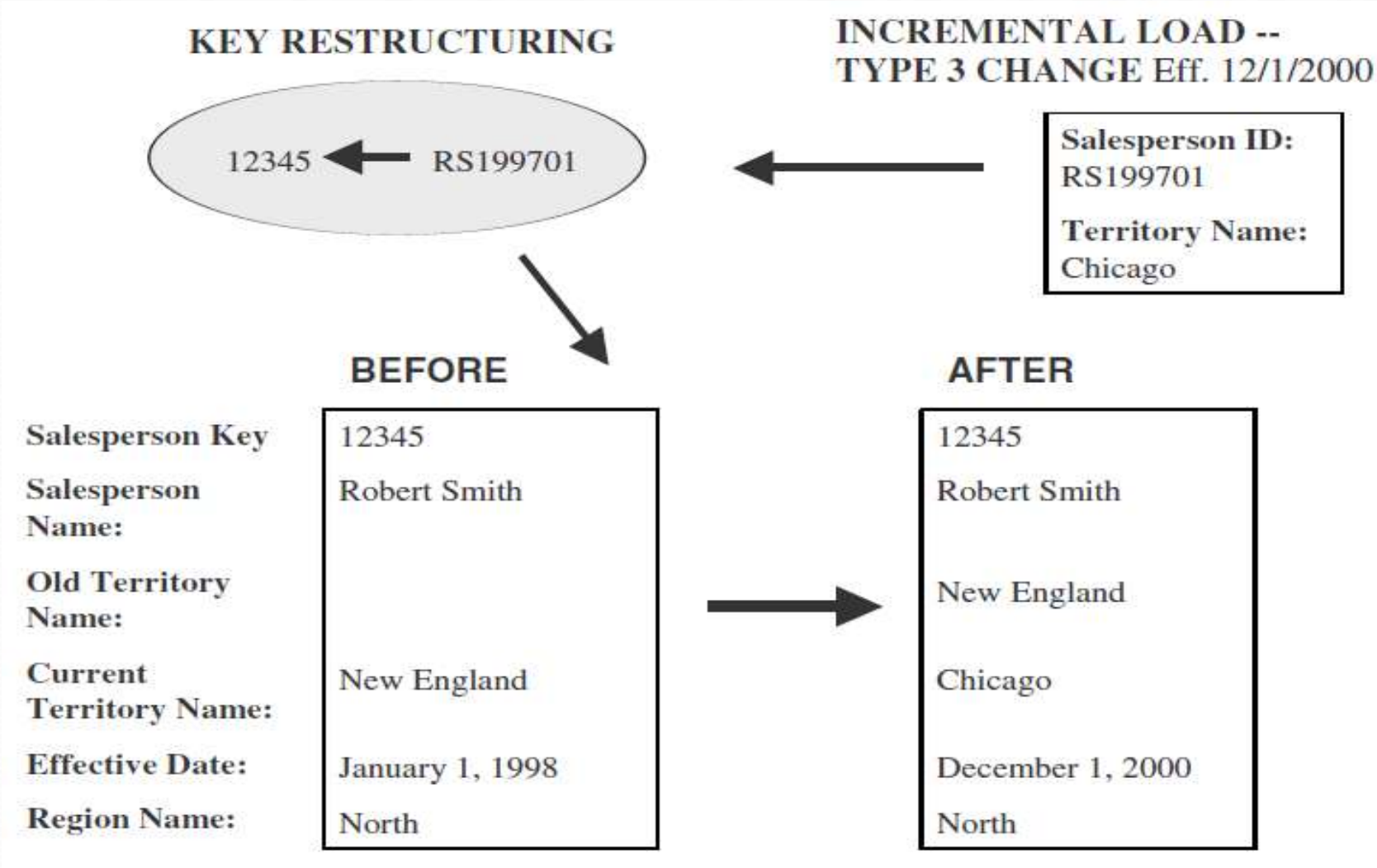
Assume your marketing department is planning a realignment of the territorial assignments for salespersons. Before making a permanent realignment, they want to count the orders in two ways: according to the current territorial alignment and also according to the proposed realignment. This type of provisional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



Example:

- Show updates to the following dimension table. Consider Type1, 2, 3 Changes wherever applicable.
- Customer (customerId, Name, Code, Marital status, Address, State, Zip).
- Previous Name : Kriti Gupta . Change the previous Name to Kriti Sharma.
- Change the Marital status of the above customer from single to Married from the date Oct 1 2016.
- Change address of the above customer from New England to Chicago from 1 Dec 201

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
- Rapidly changing dimensions
- Junk dimensions

Large dimensions

- 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.

Large dimensions

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

Customer

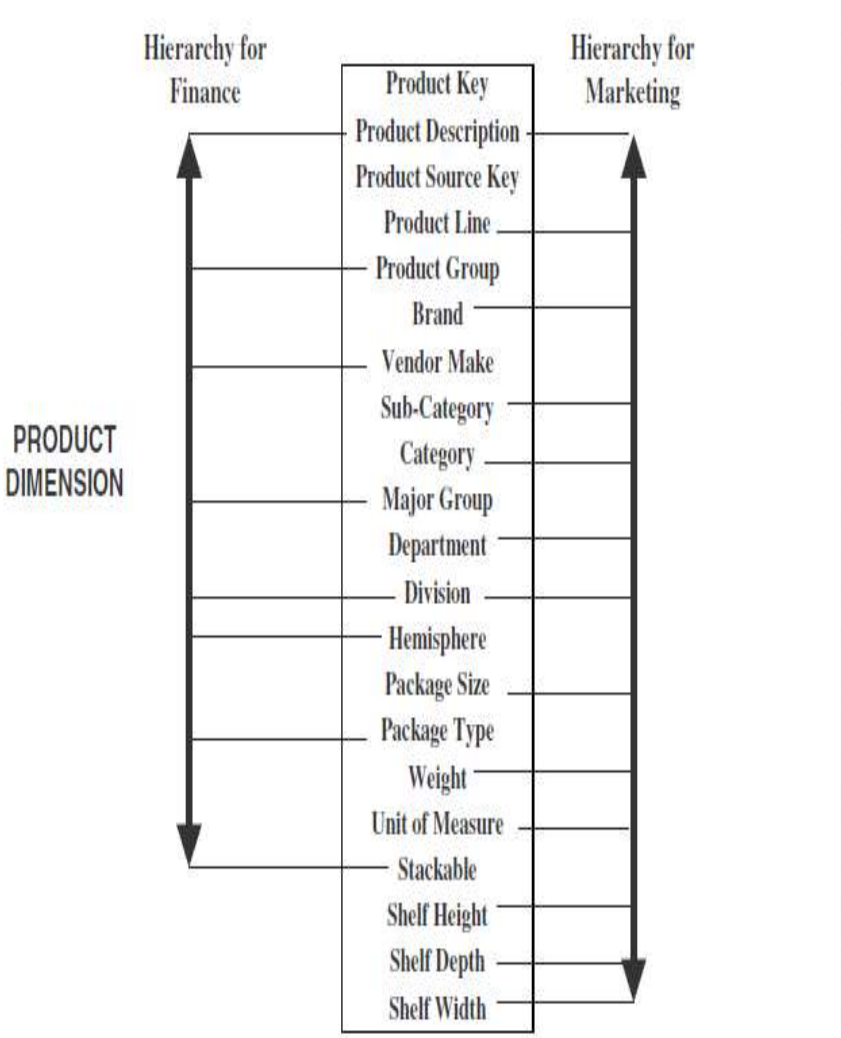
- Huge—in the range of 20 million rows
- Easily up to 150 dimension attributes
- Can have multiple hierarchies

Product

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

Multiple Hierarchies

- One set of attributes may form the hierarchy for the marketing department. The finance department may need to use their own set of attributes from the same product dimension to drill down or up.



Rapidly changing dimensions

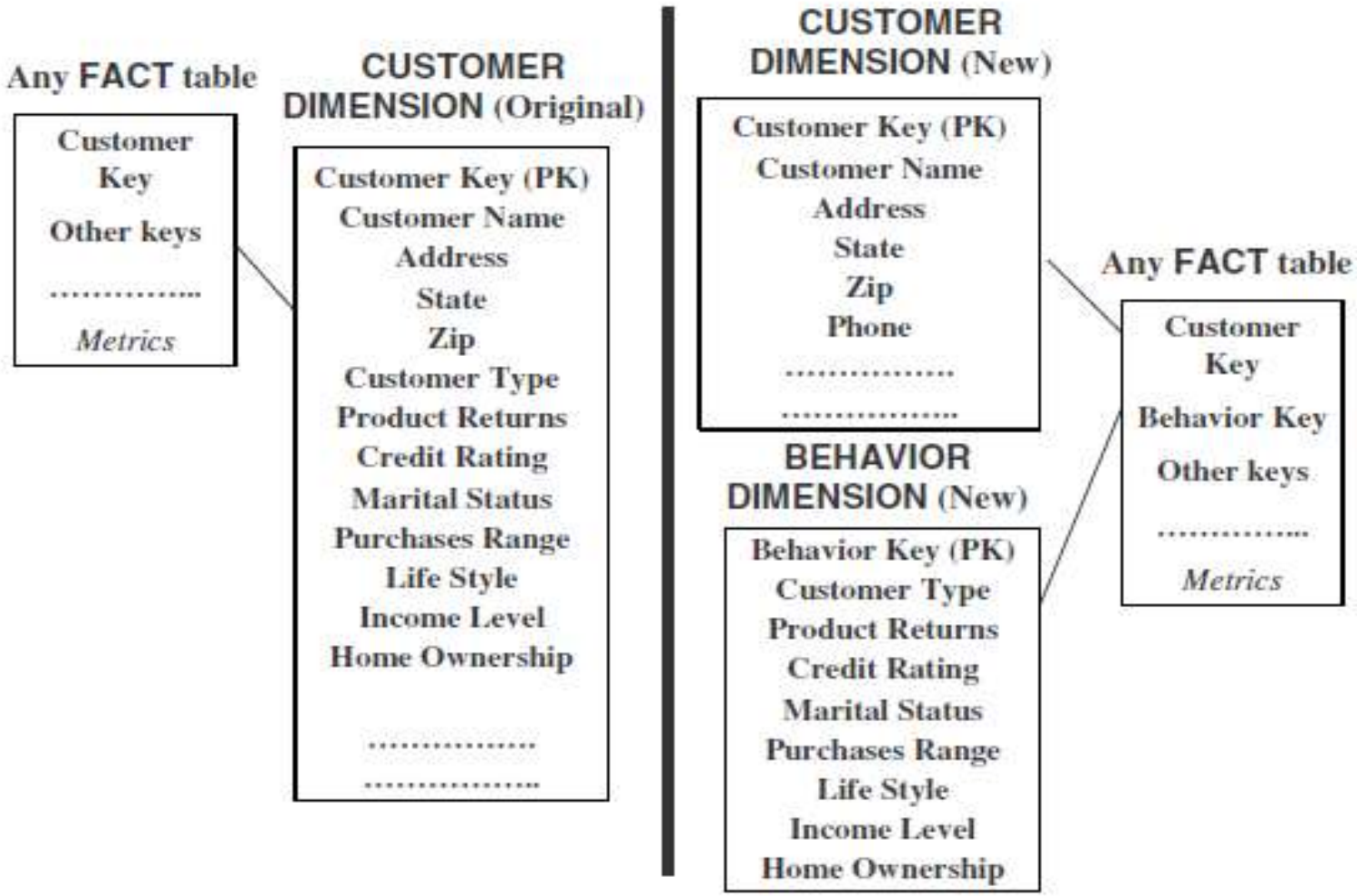
We had assumed that dimension attributes do not change too rapidly. If the change is a Type 2 change, you know that you have to create another row with the new value of the attribute. If the value of the attribute changes again, then you create another row with the newer value. What if the value changes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

Rapidly changing dimensions

Break the large dimension table into one or more simpler dimension tables.

- You need to break off the rapidly changing attributes into another dimension table, leaving the slowly changing attributes behind in the original table.

Rapidly changing dimensions



Junk Dimensions

A Junk Dimension is a dimension table consisting of attributes that do not belong in the fact table or in any of the existing dimension tables. The nature of these attributes is usually text or various flags, e.g. non-generic comments or just simple yes/no or true/false indicators. These kinds of attributes are typically remaining when all the obvious dimensions in the business process have been identified and thus the designer is faced with the challenge of where to put these attributes that do not belong in the other dimensions.

Junk Dimensions

Here are the main choices:

- Exclude and discard all flags and texts. Obviously, this is not a good option for the simple reason that you are likely to throw away some useful information.
- Place the flags and texts unchanged in the fact table. This option is likely to swell up the fact table to no specific advantage.
- Make each flag and text a separate dimension table on its own. Using this option, the number of dimension tables will greatly increase.
- Keep only those flags and texts that are meaningful; group all the useful flags into a single “junk” dimension. “Junk” dimension attributes are useful for constraining queries based on flag/text values.

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.

Junk Dimensions

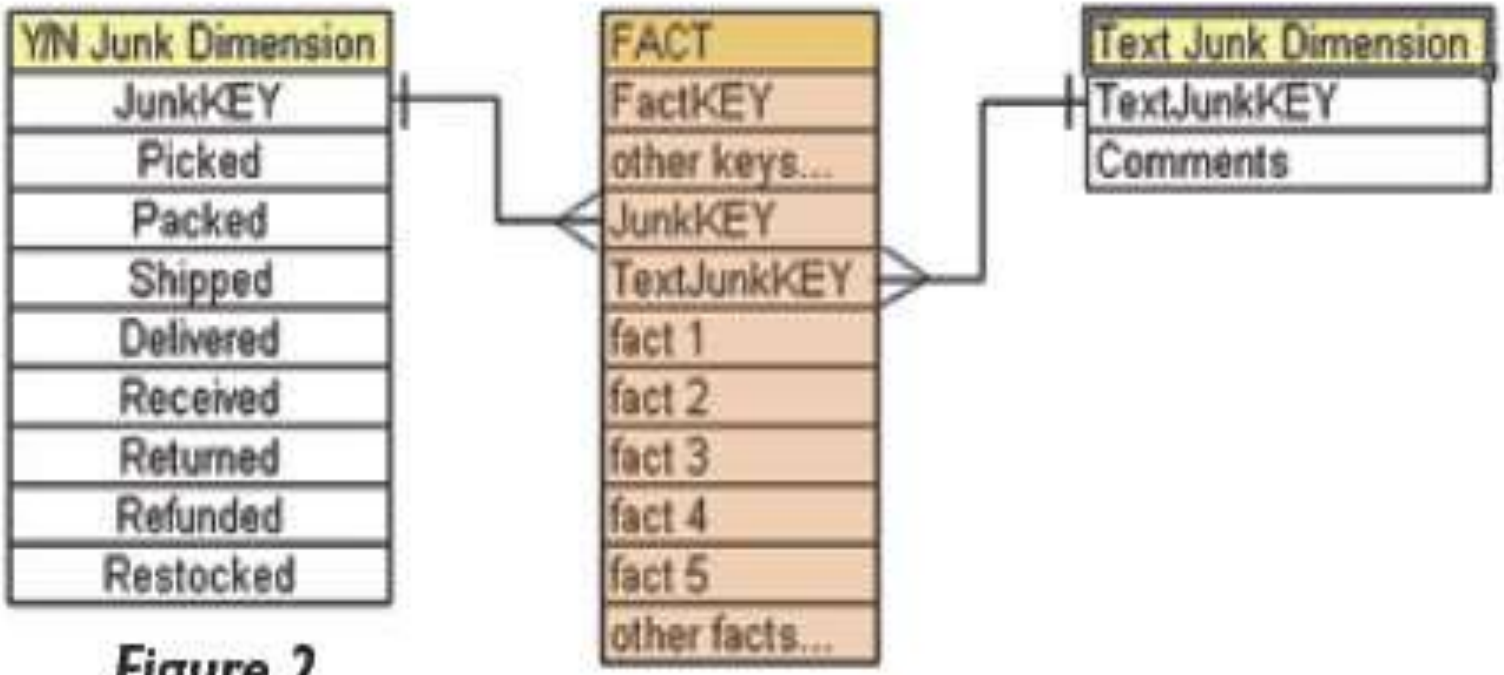
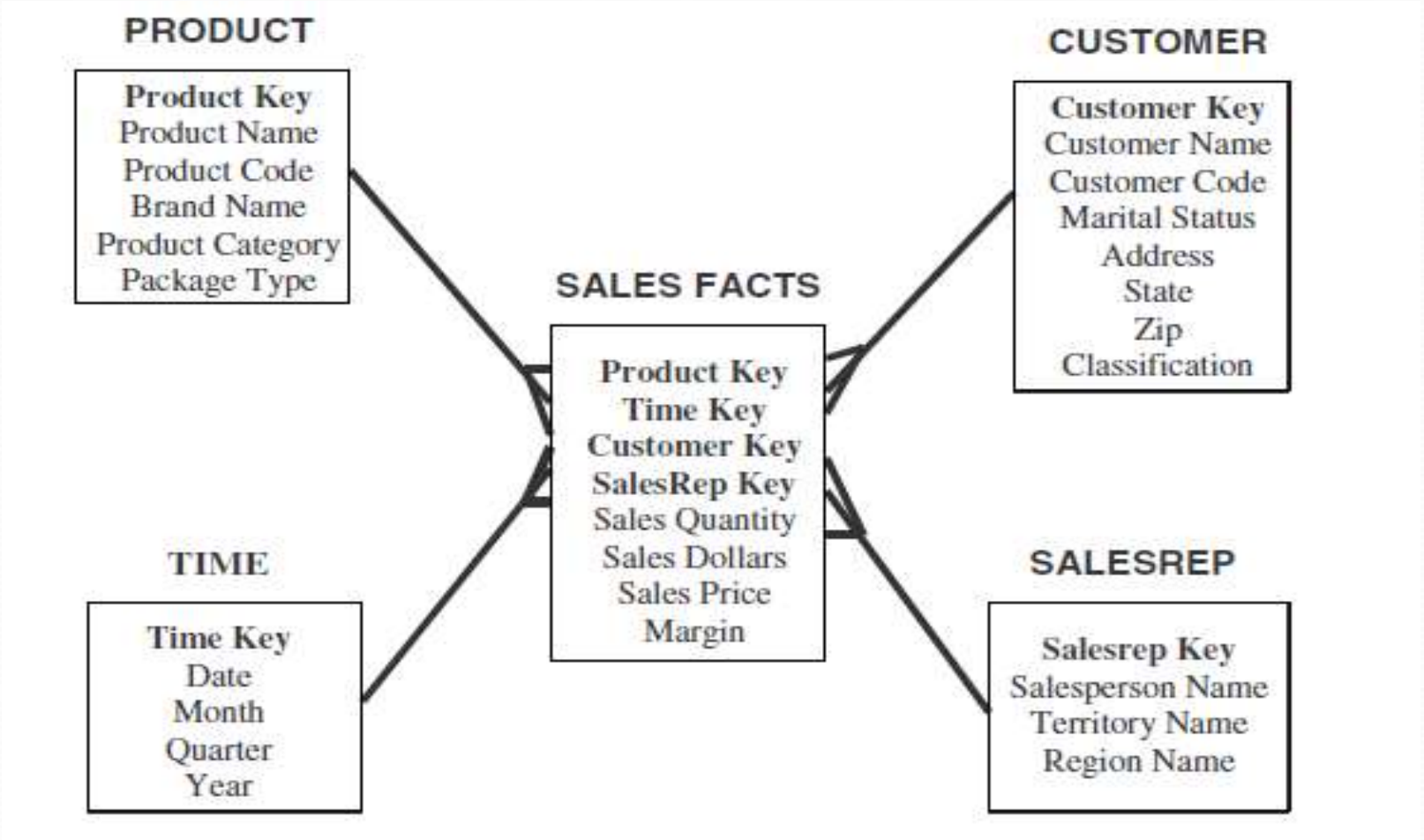


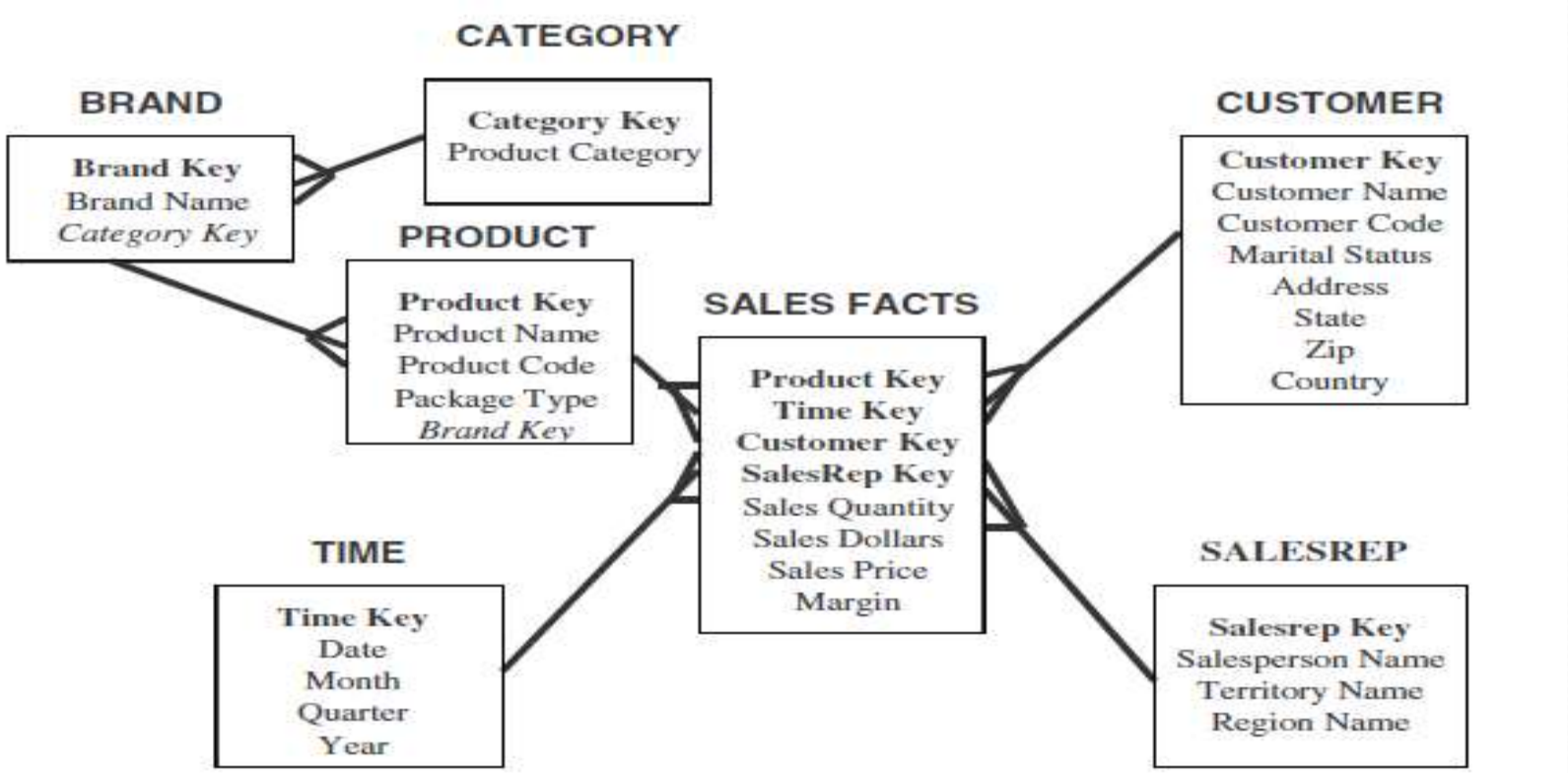
Figure 2

Associating junk dimensions with the FACT table

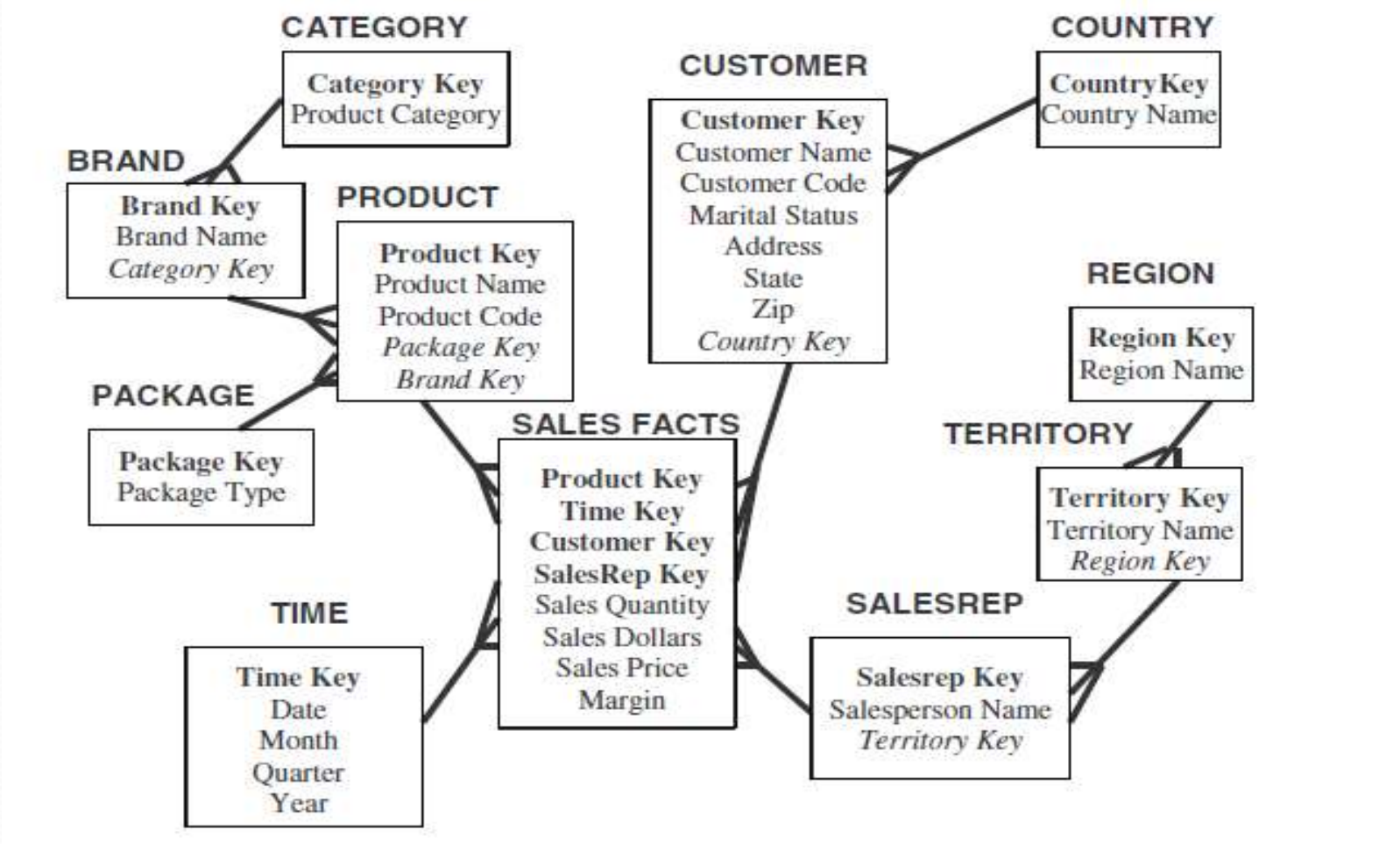
Star schema for sales



Normalized product dimension



Sales snowflake schema

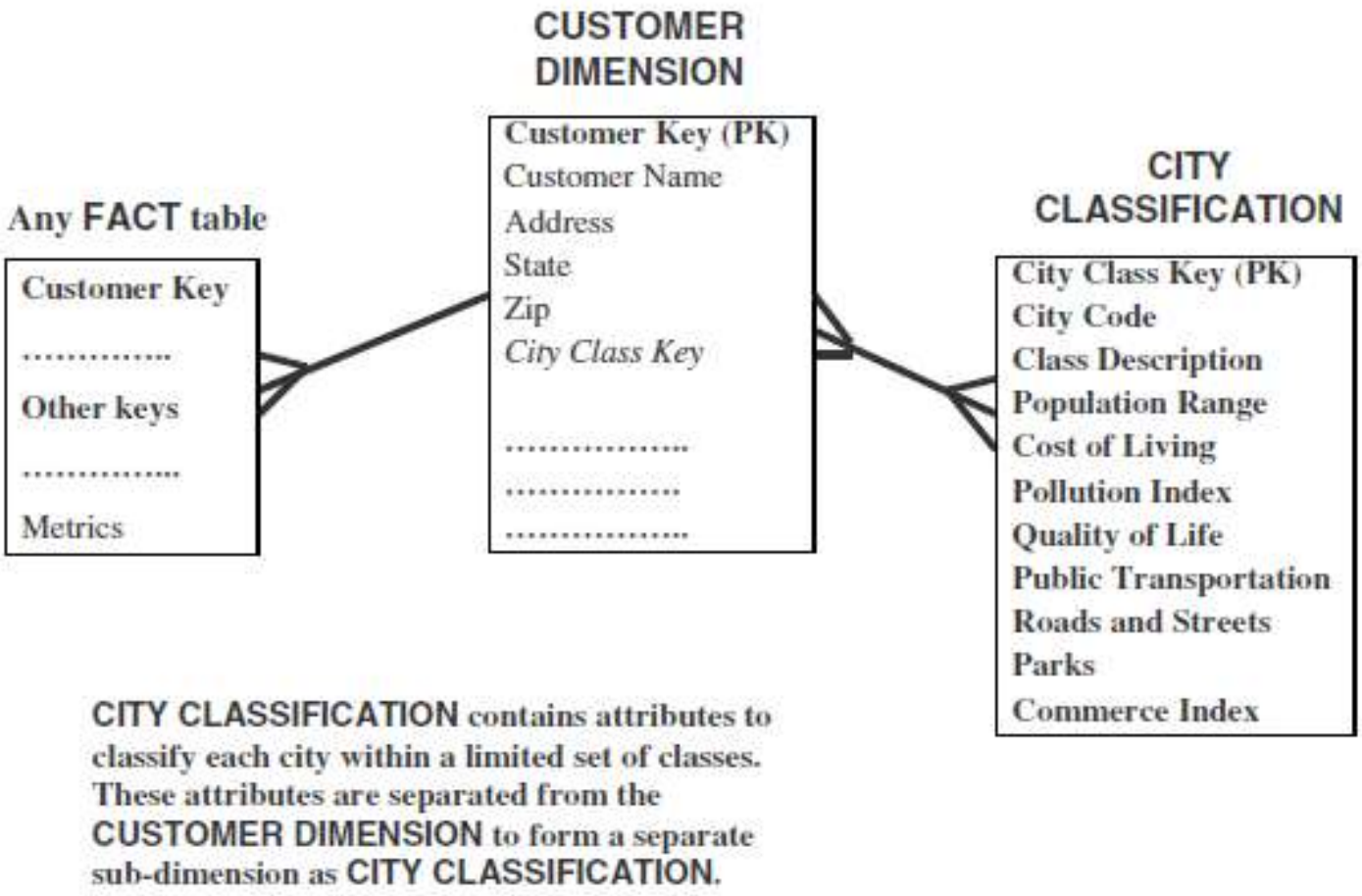


Advantages and disadvantages

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

Snowflaking is not generally recommended in a data warehouse environment. Query performance takes the highest significance in a data warehouse and snowflaking hampers the performance.

When to snowflake



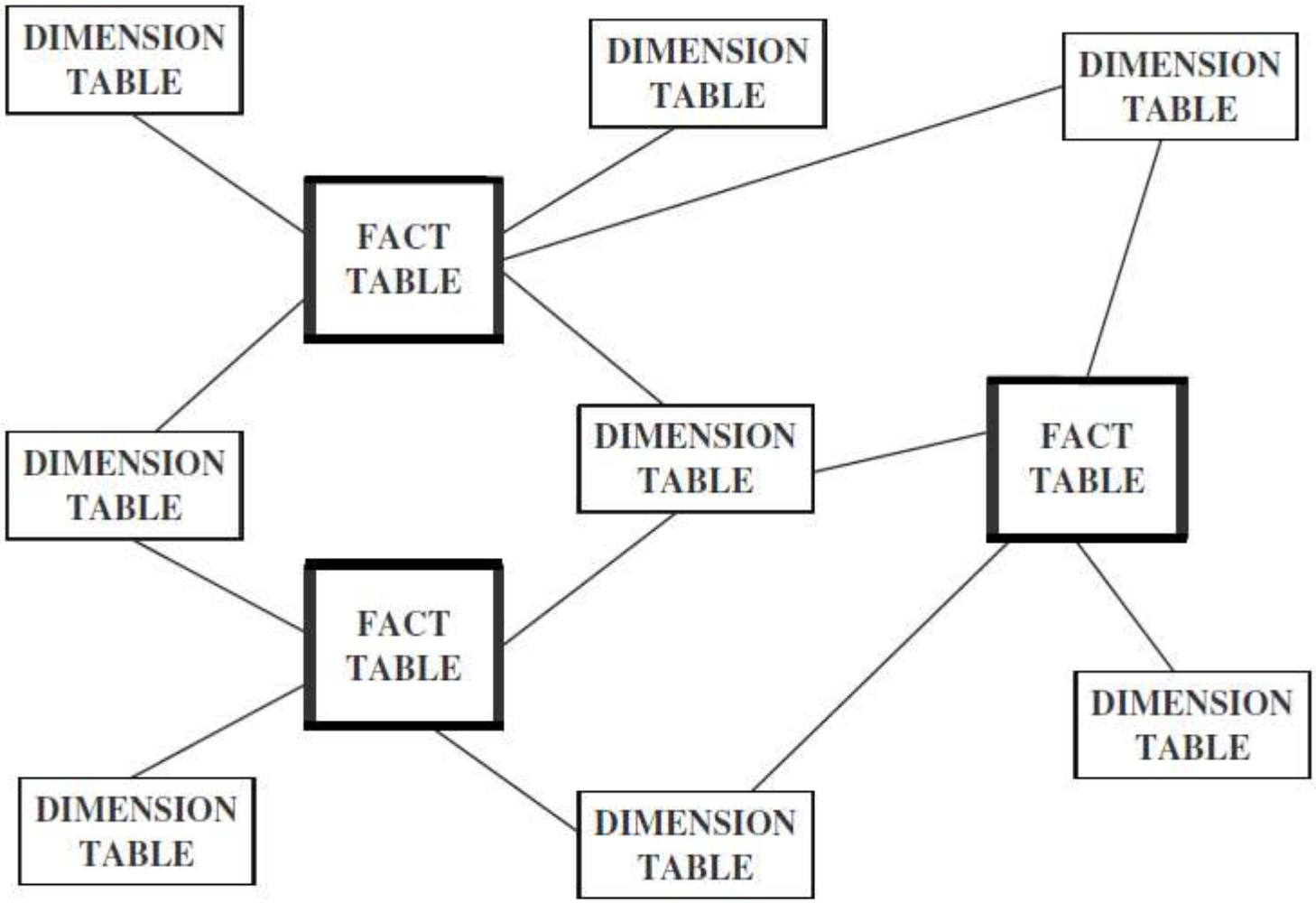
When to snowflake

- It makes a lot of sense to separate out the demographic attributes into another table.
- You usually load the demographic data at different times from the times for the load of the other dimension attributes.
- Another valid reason for separating out the demographic attributes relates to the browsing of attributes.
- Users may browse the demographic attributes more than the others in the customer dimension table.

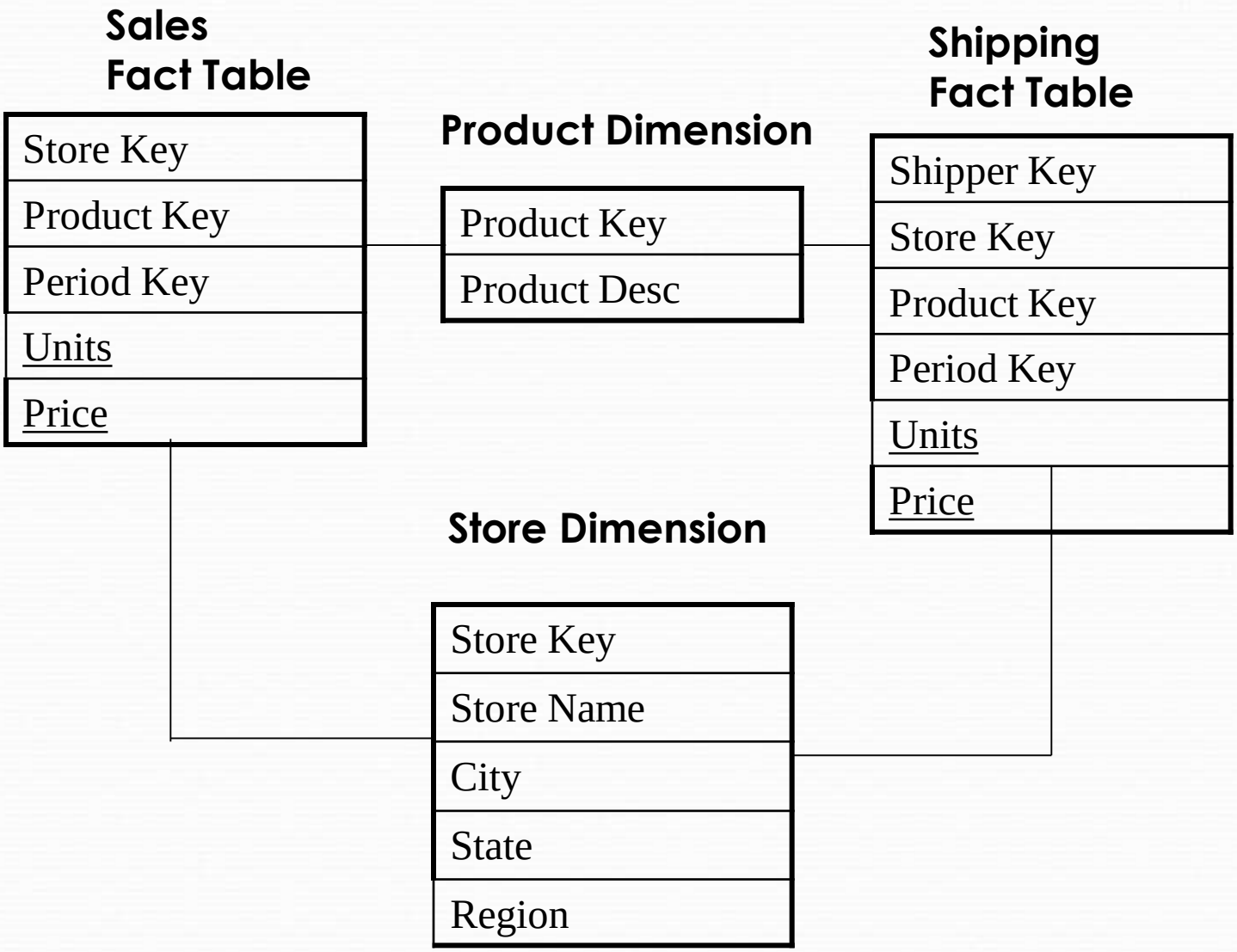
Fact Constellation

- Multiple fact tables share dimension tables.
- This schema is viewed as collection of stars hence called galaxy schema or fact constellation.
- Each STAR serves a specific purpose to track the measures stored in the fact table. When you have a collection of related STAR schemas, you may call the collection a family of STARS.
- Families of STARS are formed for various reasons. You may form a family by just adding aggregate fact tables and the derived dimension tables to support the aggregates.
- Sometimes, you may create a core fact table containing facts interesting to most users and customized fact tables for specific user groups. Many factors lead to the existence of families of STARS.

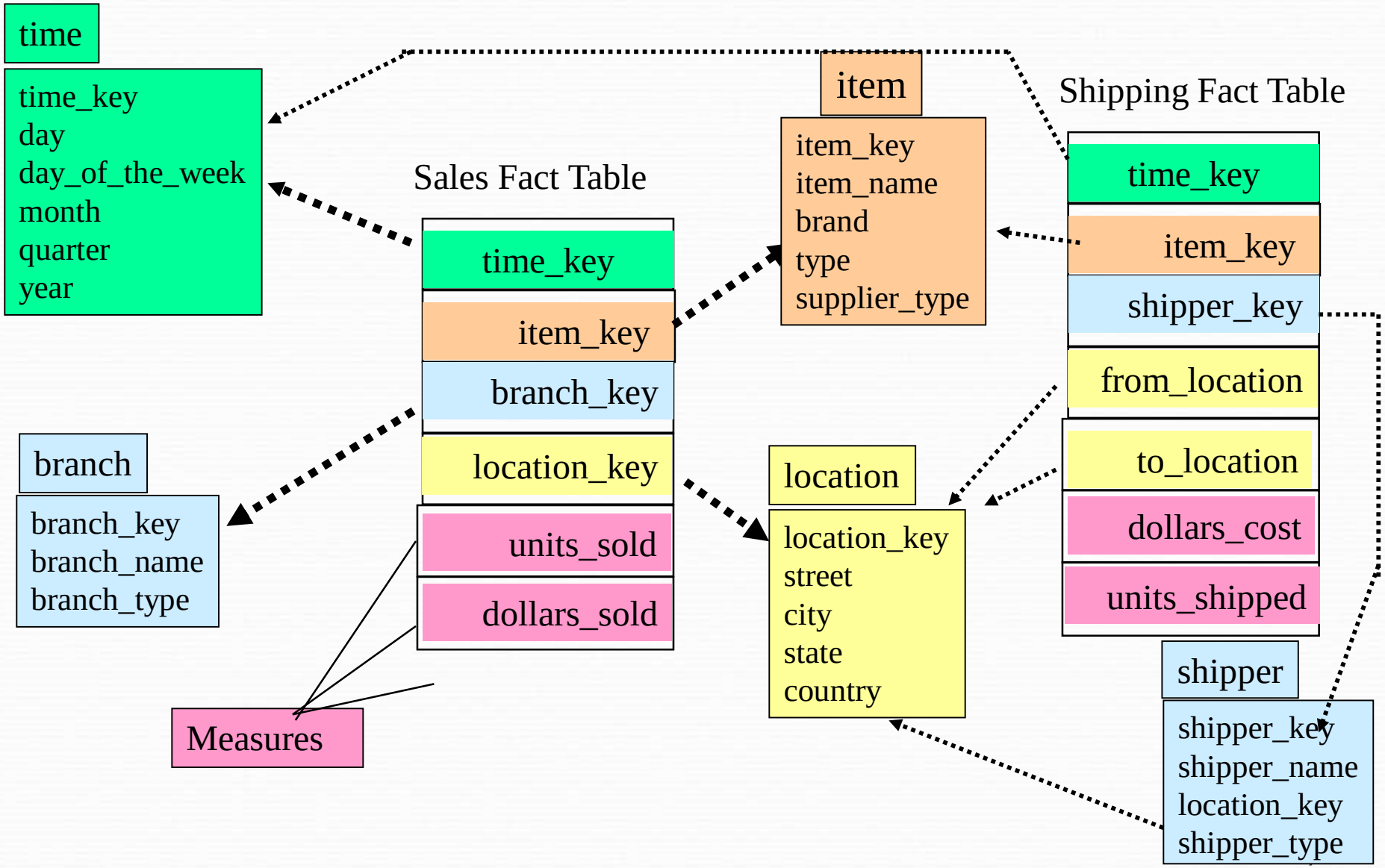
Fact Constellation: Family of STARS.



Fact Constellation



Example of Fact Constellation



Exercise Question -2 “A hotel chain”

Give information package for recording information requirement for "Hotel Occupancy“ considering dimensions like time, Hotel etc. Design star schema from information package.

Information package: hotel occupancy

Information Subject: Hotel Occupancy

Dimensions

Time	Hotel	Room Type			
Year	Hotel Line	Room Type			
Quarter	Branch Name	Room Size			
Month	Branch Code	Number of Beds			
Date	Region	Type of Bed			
Day of Week	Address	Max. Occupants			
Day of Month	City/State/ Zip	Suite			
Holiday Flag	Construc-tion Year	Refrige-rator			
	Renova-tion Year	Kichen-nette			

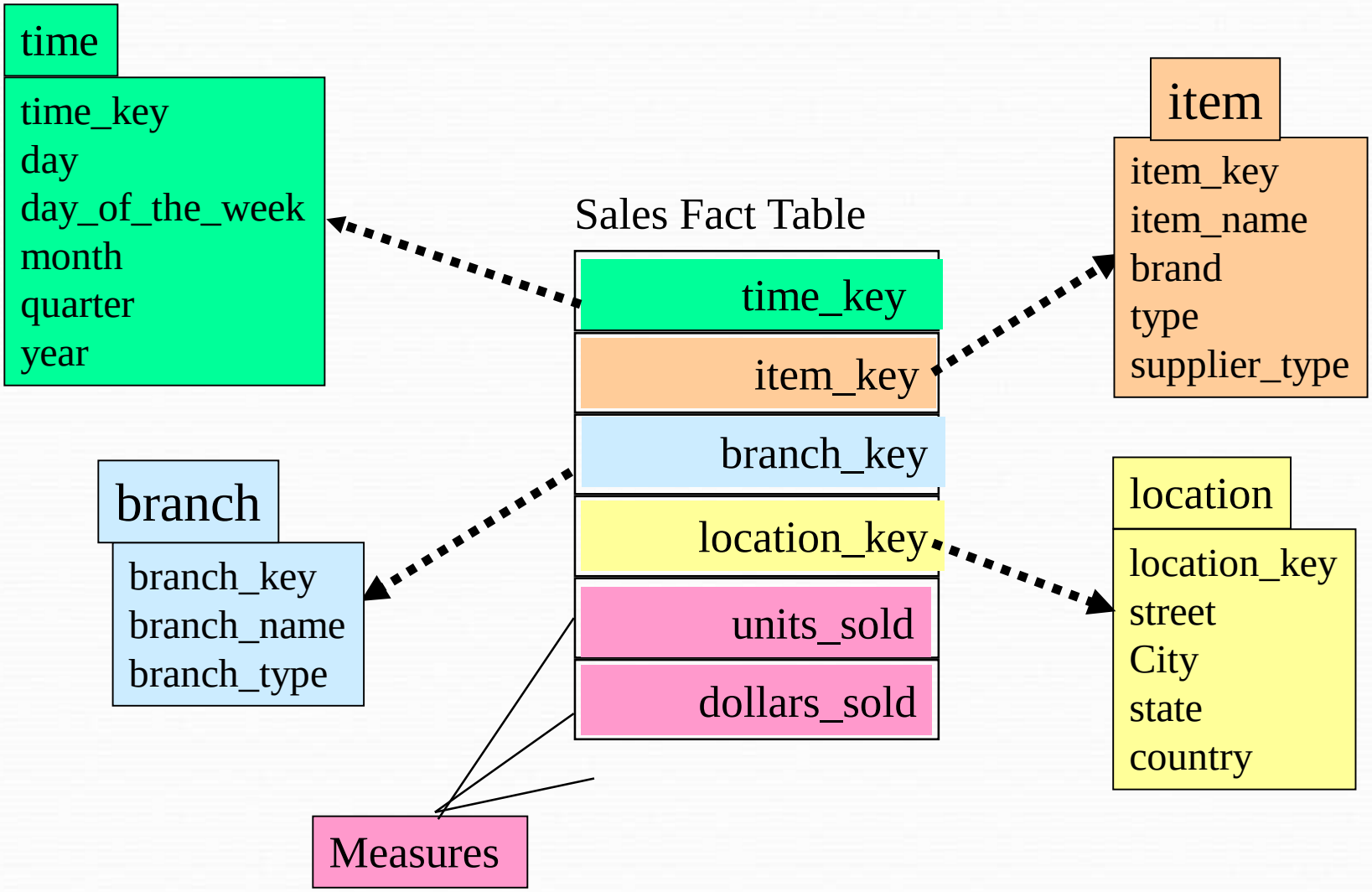
Facts: Occupied Rooms, Vacant Rooms, Unavailable Rooms, Number of Occupants, Revenue

Hierarchies / Categories

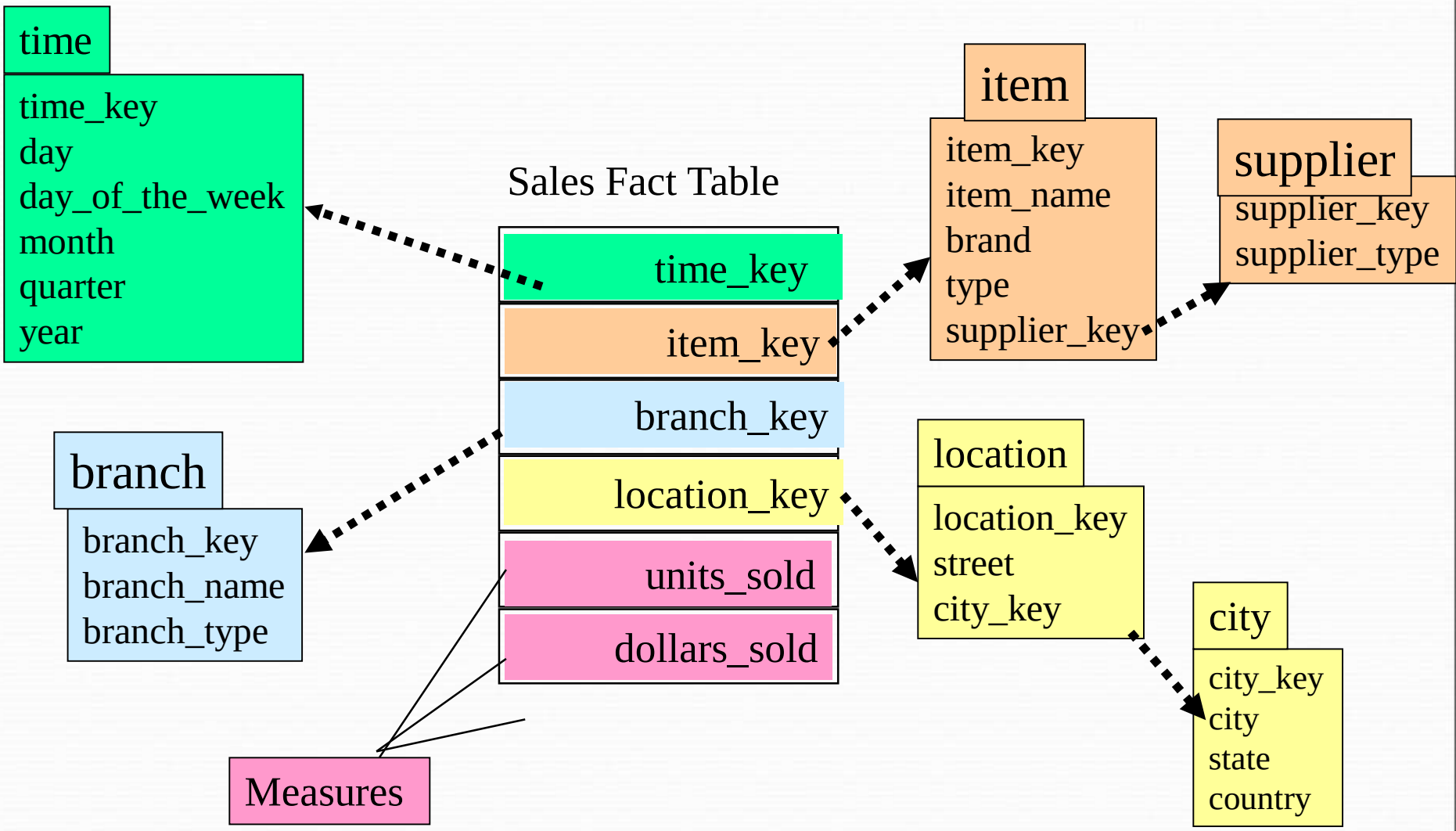
Example of Star Schema

- A star schema for *AllElectronics* sales company is shown in Figure. Sales are considered along four dimensions namely, *time, item, branch*, and *location*.
- The schema contains a central fact table for *sales* that contains keys to each of the four dimensions, along with two measures: *dollars sold* and *units sold*.

Example of Star Schema



Example of Snowflake Schema



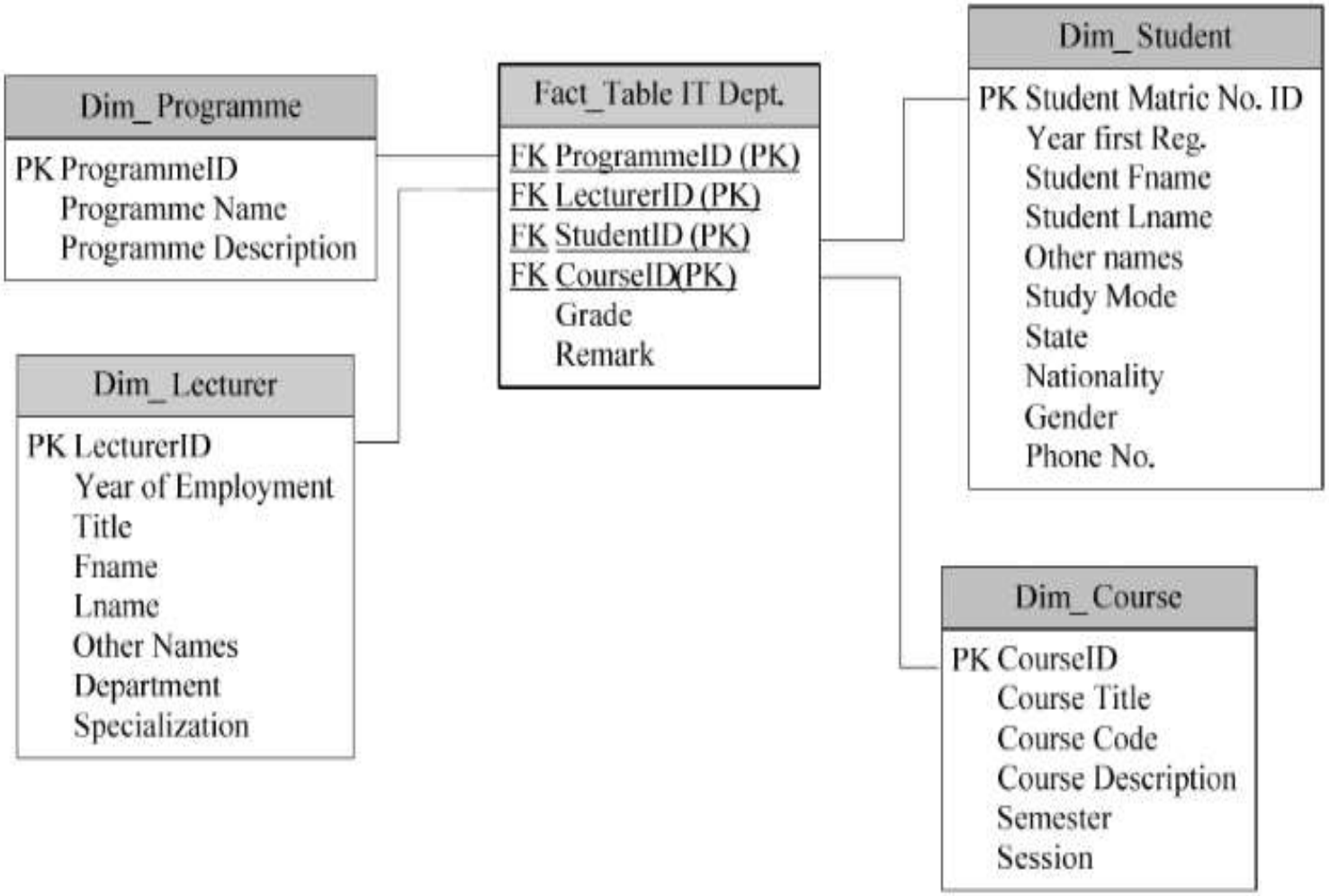
Example

Q1. (A) A bank wants to develop a data warehouse for effective decision-making about their loan schemes. The bank provides loans to customers for various purposes like House Building Loan, Car Loan, Educational Loan, Personal Loan, etc. The whole country is categorized into a number of regions, namely, North, South, East and West. Each region consists of a set of states. Loan is disbursed to customers at interest rates that change from time to time. Also, at any given point of time, the different types of loans have different rates. The data warehouse should record an entry for each disbursement of loan to customer. With respect to the above business scenario,

(1) Design an information package diagram. Clearly explain all aspects of the diagram (5)

(2) Draw a star schema for the data warehouse clearly identifying the Fact table(s), Dimension table(s), their attributes and measures. (5)

Design data warehouse for "College Admission System“. Identify different dimensions and measures. Design Star and Snowflake Schema from information package.


Dim_Programme
PK ProgrammeID
Programme Name
Programme Description
Dim_Lecturer
PK LecturerID
Year of Employment
Title
Fname
Lname
Other Names
Department
Specialization
Fact_Table IT Dept.
FK ProgrammeID (PK)
FK LecturerID (PK)
FK StudentID (PK)
FK CourseID(PK)
Grade
Remark
Dim_Student
PK Student Matric No. ID
Year first Reg.
Student Fname
Student Lname
Other names
Study Mode
State
Nationality
Gender
Phone No.
Dim_Course
PK CourseID
Course Title
Course Code
Course Description
Semester
Session

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.

Aggregate fact tables

Query 1: Total sales for customer number 12345678 during the first week of December 2000 for product Widget-1.

Query 2: Total sales for customer number 12345678 during the first three months of 2000 for product Widget-1.

Query 3: Total sales for all customers in the South-Central territory for the first two quarters of 2000 for product category Bigtools.

Scrutinize these queries and determine how the totals will be calculated in each case.

The totals will be calculated by adding the sales quantities and sales dollars from the qualifying rows of the fact table. In each case, let us review the qualifying rows that contribute to the total in the result set.

Aggregate fact tables

Query 1: All fact table rows where the customer key relates to customer number 12345678, the product key relates to product Widget-1, and the time key relates to the seven days in the first week of December 2000. Assuming that a customer may make at most one purchase of a single product in a single day, only a maximum of 7 fact table rows participate in the summation.

Query 2: All fact table rows where the customer key relates to customer number 12345678, the product key relates to product Widget-1, and the time key relates to about 90 days of the first quarter of 2000. Assuming that a customer may make at most one purchase of a single product in a single day, only about 90 fact table rows or less participate in the summation.

Query 3: All fact table rows where the customer key relates to all customers in the South-Central territory, the product key relates to all products in the product category Bigtools, and the time key relates to about 180 days in the first two quarters of 2000. In this case, clearly a large number of fact table rows participate in the summation.

Aggregate fact tables

Obviously, Query 3 will run long because of the large number of fact table rows to be retrieved. What can be done to reduce the query time? This is where aggregate tables can be helpful.

There are about two billion rows of the base fact table with the lowest level of granularity. Please study the calculations shown below:

Time dimension: 5 years × 365 days = 1825

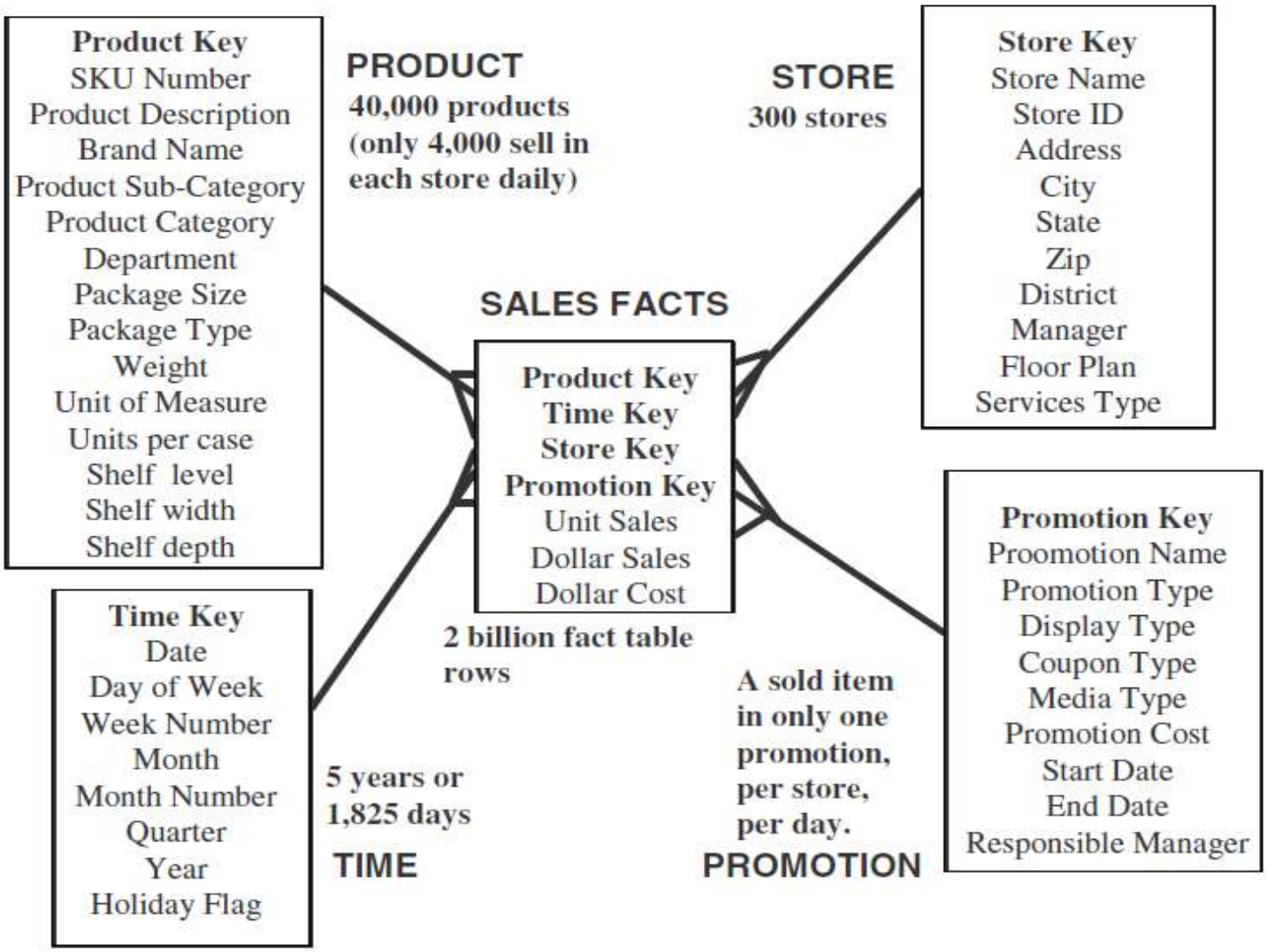
Store dimension: 300 stores reporting daily sales

Product dimension: 40,000 products in each store (about 4000 sell in each store daily)

Promotion dimension: a sold item may be in only one promotion in a store on a given day

Maximum number of base fact table records: 1825 × 300 × 4000 × 1 = 2 billion

Aggregate fact tables



Need for Aggregates

In those 300 stores, assume there are 500 products per brand. Of the 40,000 products, assume that there is at least one sale per product per store per week. Let us estimate the number of fact table rows to be retrieved and summarized for the following types of queries:

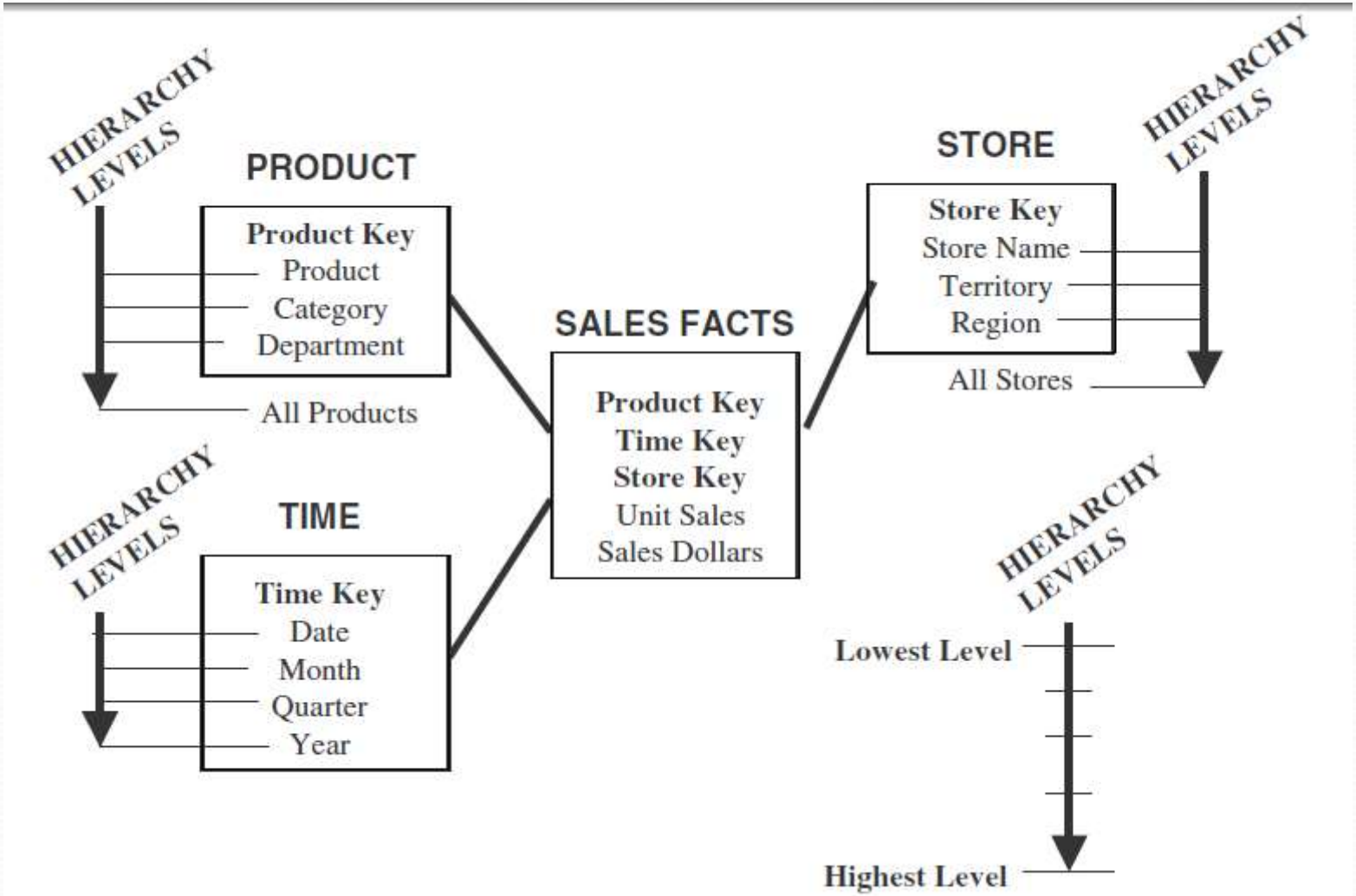
1. Query involves 1 product, 1 store, 1 week retrieve/summarize only 1 fact table row
2. Query involves 1 product, all stores, 1 week—retrieve/summarize 300 fact table rows
3. Query involves 1 brand, 1 store, 1 week—retrieve/summarize 500 fact table rows
4. Query involves 1 brand, all stores, 1 year—retrieve/summarize (500 products *300 stores* 52 weeks) 7,800,000 fact table rows

Need for Aggregates

Suppose you had pre calculated and created an aggregate fact table in which each row summarized the totals for a brand, per store, per week. Then the third query must retrieve only one row from this aggregate fact table. Similarly, the last query must retrieve only 15,600(300*52) rows from this aggregate fact table, much less than the 7 million rows. Further, if you pre calculate and create another aggregate fact table in which each row summarized the totals for a brand, per store, per year, the last query must retrieve only 300 rows.

"Aggregates have fewer rows than the base tables. Therefore, when most of the queries are run against the aggregate fact tables instead of the base fact table, you notice a tremendous boost to performance in the data warehouse. Formation of aggregate fact tables is certainly a very effective method to improve query performance."

Dimension hierarchies



Multi-Way Aggregate Fact Tables.

One-Way Aggregates: When you rise to higher levels in the hierarchy of one dimension and keep the level at the lowest in the other dimensions, you create one-way aggregate tables. Please review the following examples:

- Product category by store by date
- Product department by store by date
- All products by store by date

Multi-Way Aggregate Fact Tables.

Two-Way Aggregates : When you rise to higher levels in the hierarchies of two dimensions and keep the level at the lowest in the other dimension, you create two-way aggregate tables. Please review the following examples:

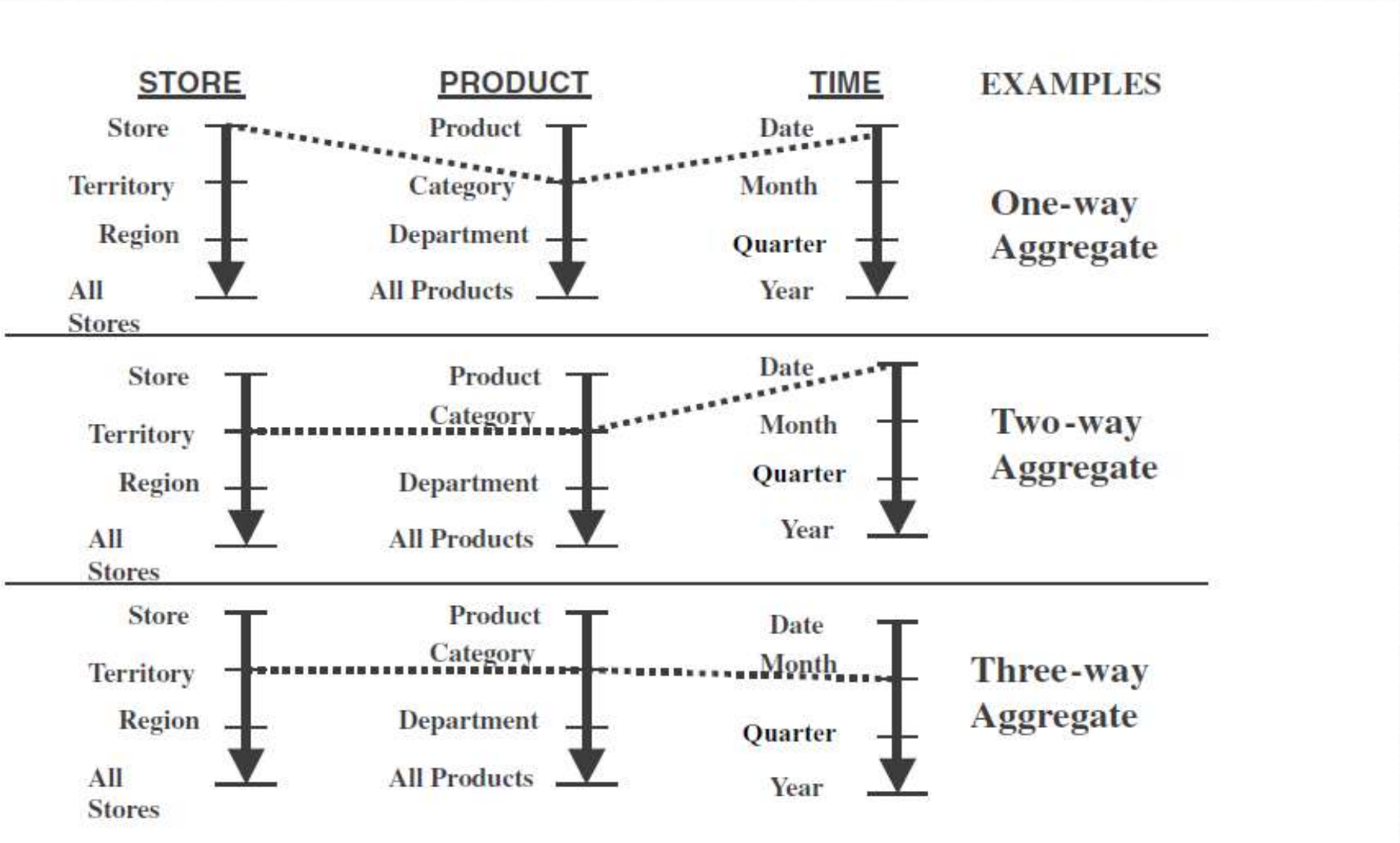
- Product category by territory by date
- Product category by region by date
- Product category by all stores by date

Multi-Way Aggregate Fact Tables.

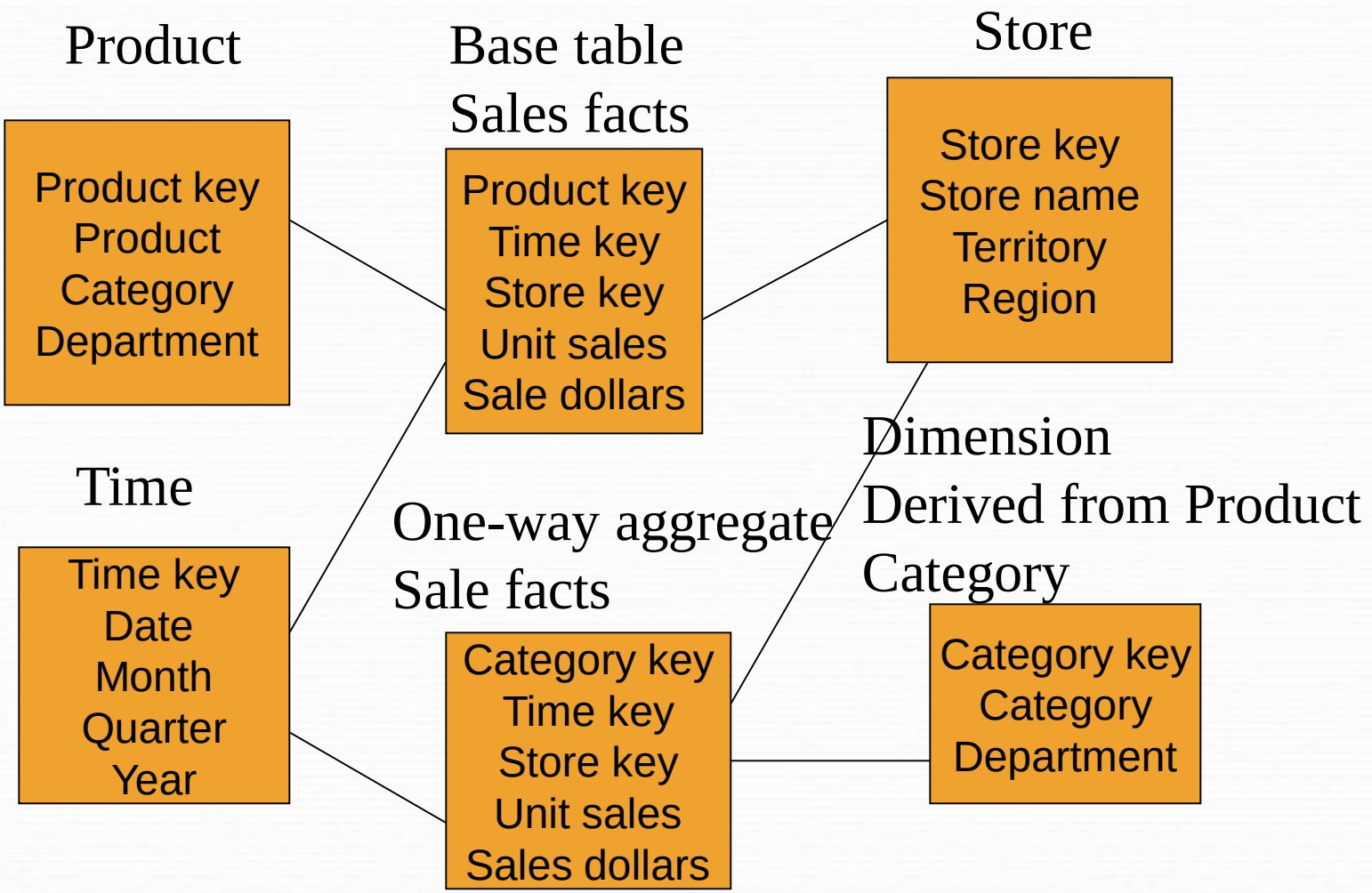
Three-Way Aggregates: When you rise to higher levels in the hierarchies of all the three dimensions, you create three-way aggregate tables. Please review the following examples:

- Product category by territory by month
- Product department by territory by month
- All products by territory by month

Multi-Way Aggregate Fact Tables



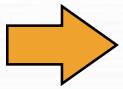
Aggregate Fact Tables



Aggregate - Example

- Add up amounts for day 1
- In SQL: SELECT sum(amt) FROM SALE
  WHERE date = 1

sale	prodId	storeId	date	amt
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4



81

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