Chapter 7
Multi Dimensional Data Modeling

Content of this presentation has been taken from Book

"Fundamentals of Business Analytics"

RN Prasad and Seema Acharya

Published by Wiley India Pvt. Ltd.

Basis

- You are already familiar with the concepts relating to basics of RDBMS, OLTP, and OLAP, role of ERP in the enterprise as well as “enterprise production environment” for IT deployment. In the previous lectures, you have been explained the concepts - Types of Digital Data, Introduction to OLTP and OLAP, Business Intelligence Basics, and Data Integration . With this background, now its time to move ahead to think about “how data is modelled”.
    - Just like a circuit diagram is to an electrical engineer,
    - an assembly diagram is to a mechanical Engineer, and
    - a blueprint of a building is to a civil engineer
    - So is the data models/data diagrams for a data architect.
- But is “data modelling” only the responsibility of a data architect?
  The answer is Business Intelligence (BI) application developer today is involved in designing, developing, deploying, supporting, and optimizing storage in the form of data warehouse/data marts.
- To be able to play his/her role efficiently, the BI application developer relies heavily on data models/data diagrams to understand the schema structure, the data, the relationships between data, etc.

In this lecture, we will learn

- About basics of data modelling
- How to go about designing a data model at the conceptual and logical levels?
- Pros and Cons of the popular modelling techniques such as ER modelling and dimensional modelling

Case Study – "TenToTen Retail Stores"

- A new range of cosmetic products has been introduced by a leading brand, which TenToTen wants to sell through its various outlets.
- In this regard TenToTen wants to study the market and the consumer's choice of cosmetic products.
- As a promotional strategy the group also wants to offer attractive introductory offers like discounts, buy one get one free, etc.
- To have a sound knowledge of the cosmetics market, TenToTen Stores has to carry out a detailed study of the buying pattern of consumers' by geography, the sales of cosmetic products by preferred brand, etc. and then decide on a strategy to promote the product. To take right decisions on various aspects of business expansion, product promotion, preferences, etc., TenToTen Stores has decided to go in for an intelligent decision support system.
- TenToTen Retail Store taken the help of "AllSolutions" (leading consulting firms of the world) .
- After studying the requirements of TenToTen Stores, AllSolutions decided on build a data warehouse application. To construct a data model that would meet the business requirements put forth by TenToTen Stores. AllSolutions identified the following concerns that need to be addressed:

  What are the entities involved in this business process and how are they related to each other?
  What tables associated with those entities must be included in the data warehouse?
  What columns have to be included into each table?
  What are the primary keys for the tables that have been identified?
  What are the relations that the tables have with each other and which is the column on which the relationship has to be made?
  What should be the column definitions for the columns that have been identified?
  What are the other constraints to be added into the tables?

Thus, AllSolutions has zeroed down on the requirements of TenToTen Stores. Now, step for building data model can be proceeded.

Recap of some basics of Data Modelling-

- ✓ Entity
- ✓ Attribute
- ✓ Cardinality of Relationship

Data Model

- ✓ A data model is a diagrammatic representation of the data and the relationship between its different entities. It assists in identifying how the entities are related through a visual representation of their relationships and thus helps reduce possible errors in the database design. It helps in building a robust database/data warehouse.

Types of Data Model

- ✓ Conceptual Data Model
- ✓ Logical Data Model
- ✓ Physical Data Model

Conceptual Data Model

The conceptual data model is designed by identifying the various entities and the highest-level relationships between them as per the given requirements.

Let us look at some features of a conceptual data mpdel-

- It identifies the most important entities.
- It identifies relationships between different entities.
- It does not support the specification of attributes.
- It does not support the specification of the primary key.

Going back to the requirement specification of TenToTen Stores, let us design the conceptual data model (Next Slide).

In this case, the entities can be identified as

- Category (to store the category details of products).
- SubCategory (to store the details of sub-categories that belong to different categories)
- Product (to store product details).
- PromotionOffer (to store various promotion offers introduced by the company to sell products)
- ProductOffer (to map the promotion offer to a product).
- Date (to keep track of the sale date and also to analyze sales in different time periods)
- Territory (to store various territories where the stores are located).
- MarketType (to store details of various market setups, viz. “Hypermarkets &“Traditional Supermarket”, “Dollar Store”, and “Super Warehouse”).
- OperatorType (to store the details of types of operator, viz. company-operated or franchise)
- Outlet (to store the details of various stores distributed over various locations).
- Sales (to store all the daily transactions made at various stores)

Logical Data Model

The logical data model is used to describe data in as much detail as possible. While describing the data, no consideration is given to the physical implementation aspect.

Let us look at some features of a logical data model:

- It identifies all entities and the relationships among them.
- It identifies all the attributes for each entity.
- It specifies the primary key for each entity.
- It specifies the foreign keys (keys identifying the relationship between different entities).
- Normalization of entities is performed at this stage.

Normalization:

1NF
2NF
3NF and soon

Outcome of Logical Data Model

Category

Columns	*Description*	*Constraint*
ProductCategoryID	The ProductCategory ID is used to uniquely identify different types of categories.	**Primary Key**
CategoryName	The name of the corresponding category.	

SubCategory

Columns	*Description*	*Constraint*
ProductSubCategoryID	The Product SubCategory ID uniquely identifies the sub-categories that belong to each product.	**Primary Key**
SubCategoryName	The name of the corresponding sub-category.	
ProductCaregoryID	The Product Category ID to which the sub-category belongs.	Refers to the Category (ProductCategoryID)

Product

Columns	*Description*	*Constraint*
ProductID	The product code for the respective product, and will be the primary key for the Product Table.	**Primary Key**
ProductName	The name of the product.	
ProductDescription	Gives a brief description about the product.	
SubCategoryID	Describes the sub-category the product belongs to.	
DateOfProduction	Date when the corresponding product was manufactured.	"dd/mm/yyyy" format
LastDate OfPurchase	The last date the shipping was made to the warehouse.	
CurrentInventoryLevel	The number of products that are currently present in the inventory of the product.	
StandardCost	The standard price for the product.	
ListPrice	The listed price for the product.	

Outcome of Logical Data Model

Columns	Description	Constraint
Discontinued	Used to find whether the manufacturing of a particular product has been discontinued.	"Y" or "N"
DiscontinuedDate	The date when the product manufacture was discontinued.	"dd/mm/yyyy" format

PromotionOffers

Columns	Description	Constraint
PromotionOfferID	It is used to uniquely identify the various promotion offers.	**Primary Key**
PromotionType	The type of offers given.	(Discounts, Buy One Get One Free, etc.)
DiscountPercentage	The percentage of discount given on the List Price of the product.	
ComplimentaryProduct	Complimentary products that might be offered with a product.	
DateOfOfferExpiry	The date when the offer will expire.	"dd/mm/yyyy" format

ProductOffer

Columns	Description	Constraint
ProductID	Refers to a product from the product table to which the offer is to be made.	**Primary Key,** Refers to Product (ProductID)
PromotionOfferID	Refers to the promotion offer that is to be given with a product.	**Primary Key,** Refers to PromotionOffers (PromotionOfferID)

Date

Columns	Description	Constraint
DateID	The Date ID for each date for uniquely identifying each date.	**Primary Key**
Date	The corresponding date.	"dd/mm/yyyy" format
DayOfWeek	The name of the day of the week.	(Monday, Tuesday, Wednesday, Thursday, Friday, Sunday)
WeekOfMonth	The week number w.r.t. the month.	(1, 2, 3, 4)

(Continued)

Outcome of Logical Data Model

(Continued)

Columns	*Description*	*Constraint*
WeekOfYear	Describes the category the product belongs to.	(1, 2, 3, ..., 52)
MonthName	Contains the month name of the year.	(January, February, ..., December)
MonthOfYear	The month number in the year.	(1, 2, 3, ..., 12)
Quarter	The quarter period of the corresponding year.	(1, 2, 3, 4)
Year	Contains the corresponding year.	"yyyy" format

MarketType

Columns	*Description*	*Constraint*
MarketID	Uniquely identifies the store setup.	**Primary Key**
Market_Name	The store setup based on which the store operates.	(Hypermarkets & Supermarkets, Traditional Supermarket, Dollar Store, Super Warehouse)

Operator_Type

Columns	*Description*	*Constraint*
Operator_ID	Identifies the type of operation.	**Primary Key**
Operator	The type of working of the outlet/store.	(Company Operated, Franchises, etc.)

Outlets

Columns	*Description*	*Constraint*
Store ID	Each store has a respective Store ID. It will be used to uniquely identify various stores.	**Primary Key**
MarketID	The market type of the store.	Refers to the Market (MarketID)
OperatorID	The working type of the store.	Refers to the Operator_Type (Operator_ID)
TerritoryID	It contains the territory that the store belongs to.	Refers to the Territory Table
Opening Time	The time when the corresponding store opens.	24 hours format

Outcome of Logical Data Model

Columns	*Description*	*Constraint*
Closing Time	The time when the corresponding store closes.	24 hours format
StoreOpenDate	Contains the date when the store was opened.	"dd/mm/yyyy" format

Territory

Columns	*Description*	*Constraint*
TerritoryCode	Identifies each territory uniquely in the Territory Table.	**Primary Key**
Territory	The name of the territory corresponding to the territory code.	
City	The city in which the territory is present.	
State	The state to which the city belongs to.	

SalesTransaction

Columns	*Description*	*Constraint*
TransactionID	Identifies every sale that was made.	**Primary Key**
TransactionDate	Contains the date the sales were made.	Refers to the Time Table
StoreID	Contains the ID of the store from where the sale was made.	Refers to the Stores Table
ProductID	Contains the ID of the product that was sold.	Refers to the Product Table
QuantityOrdered	Contains the quantity of the product sold.	
TotalAmount	Contains the total amount of the sale made for the corresponding product.	SalesAmount = ItemSold * UnitPrice

Outcome of Logical Data Model

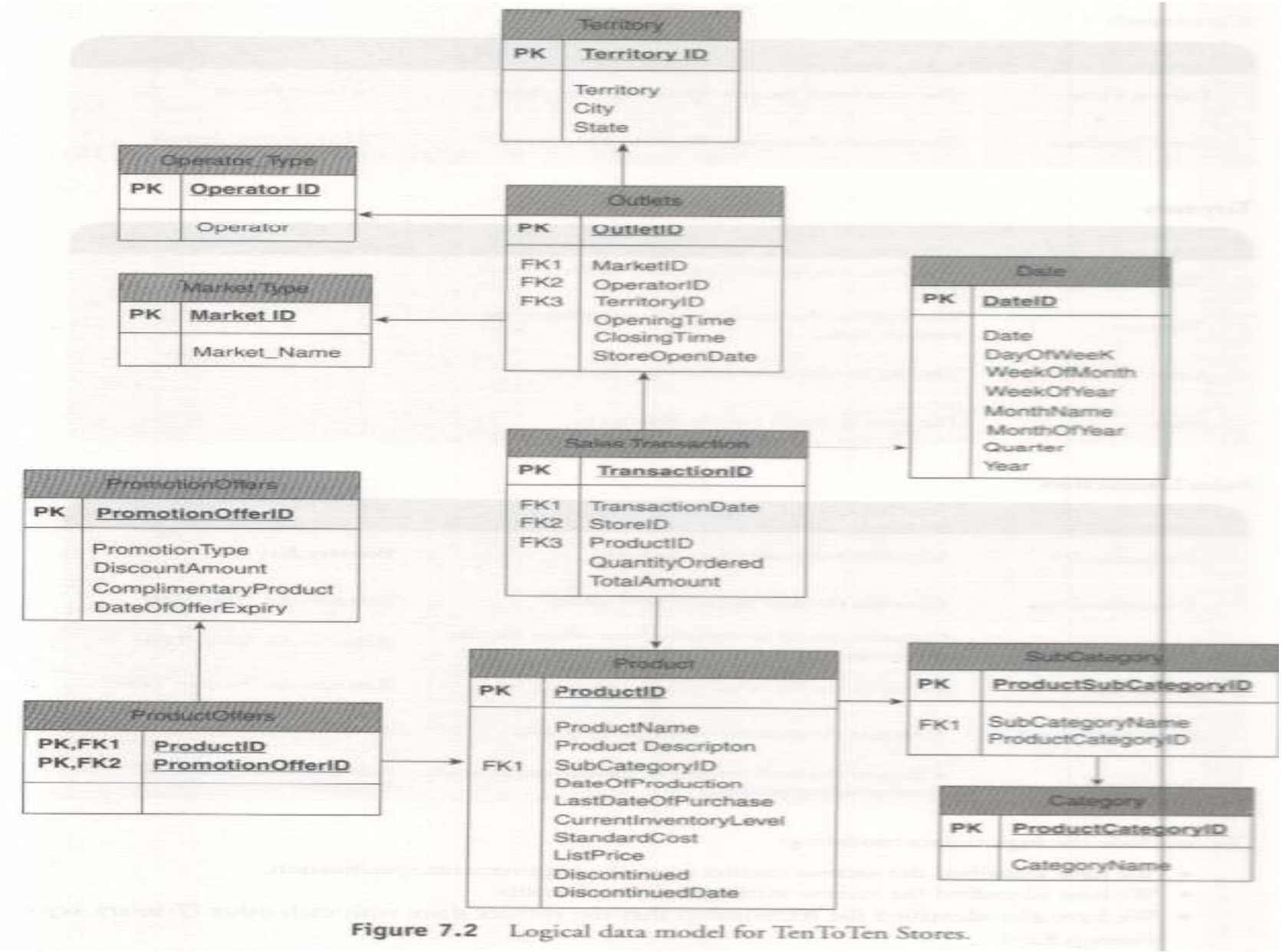


Figure 7.2 Logical data model for TenToTen Stores.

To Conclude about Conceptual Data Model

- We have identified the various entities from the requirements specification.
- We have identified the various attributes for each entity.
- We have also identified the relationship that the entities share with each other (Primary key-Foreign Key).

Compare between Logical and Conceptual Data Model

- All attributes for each entity are specified in a logical data model, whereas no attributes fre specified in a conceptual data model.
- Primary keys are present in a logical data model, whereas no primary key is present in 3 conceptual data model.
- In a logical data model, the relationships between entities are specified using primary and foreign keys, whereas in a conceptual data model, the relationships are simply without specifying attributes. It means in a conceptual data model, we only know that two related; we don't know which attributes are used for establishing the relationship between these two entities.

Physical Model

- Specification of all tables and columns.
- Foreign keys are used to identify relationships between tables.
- While logical data model is about normalization, physical data model may support de-normalization based on user requirements.
- Physical considerations (implementation concerns) may cause the physical data model to be quite different from the logical data model.
- Physical data model will be different for different RDBMS. For example, data type for a column may be different for MySQL, DB2, Oracle, SQL Server, etc.

The steps for designing a physical data model are as follows:

- Convert entities into tables/relation.
- Convert relationships into foreign keys.
- Convert attributes into columns/fields.

Outcome of Physical Data Model

SubCategory

iProductSubCategoryID	INT IDENTITY(1,1)	PRIMARY KEY
vSubCategoryName	VARCHAR(50)	NOT NULL
iProductCategoryID	INT	Category(iProductCategoryID)

Product

iProductID	INT IDENTITY(1,1)	PRIMARY KEY
vProductName	VARCHAR(50)	NOT NULL
vProductDescription	VARCHAR(250)	NOT NULL
iSubCategoryID	INT	SubCategory(iProductSubCategoryID)
dDateOfProduction	DATE	NOT NULL
dLastDateOfPurchase	DATE	NOT NULL
iCurrentInventoryLevel	INT	NOT NULL
mStandardCost	MONEY	NOT NULL
mListPrice	MONEY	NOT NULL
cDiscontinued	CHAR(1)	CHECK "Y" or "N"
dDiscontinuedDate	DATE	NULL

PromotionOffers

iPromotionOfferID	INT IDENTITY(1,1)	PRIMARY KEY
vPromotionType	VARCHAR(30)	NOT NULL
tiDiscountPercent	TINYINT	NULL
vComplimentaryProduct	VARCHAR(50)	NULL
dDateOfOfferExpiry	DATE	NOT NULL

ProductOffer

iProductID	INT	PRIMARY KEY Product (iProductID)
iPromotionOfferID	INT	PRIMARY KEY PromotionOffers (PromotionOfferID)

Date

iDateID	INT IDENTITY(1,1)	PRIMARY KEY
dDate	DATE	NOT NULL
vDayOfWeek	VARCHAR(10)	NOT NULL
iWeekOfMonth	INT	NOT NULL

(Continued)

Outcome of Phvsical Data Model

iWeekOfYear	INT	NOT NULL
vMonthName	VARCHAR(10)	NOT NULL
iMonthOfYear	INT	NOT NULL
iQuarter	INT	NOT NULL
iYear	INT	NOT NULL

MarketType

iMarketID	INT IDENTITY(1,1)	PRIMARY KEY
vMarket_Name	VARCHAR(30)	NOT NULL

Operator_Type

iOperator_ID	INT IDENTITY(1,1)	PRIMARY KEY
vOperator	VARCHAR(50)	NOT NULL

Territory

iTerritoryCode	INT IDENTITY(1,1)	PRIMARY KEY
vTerritory	VARCHAR(30)	NOT NULL
vCity	VARCHAR(6)	NOT NULL
vState	VARCHAR(6)	NOT NULL

Outlets

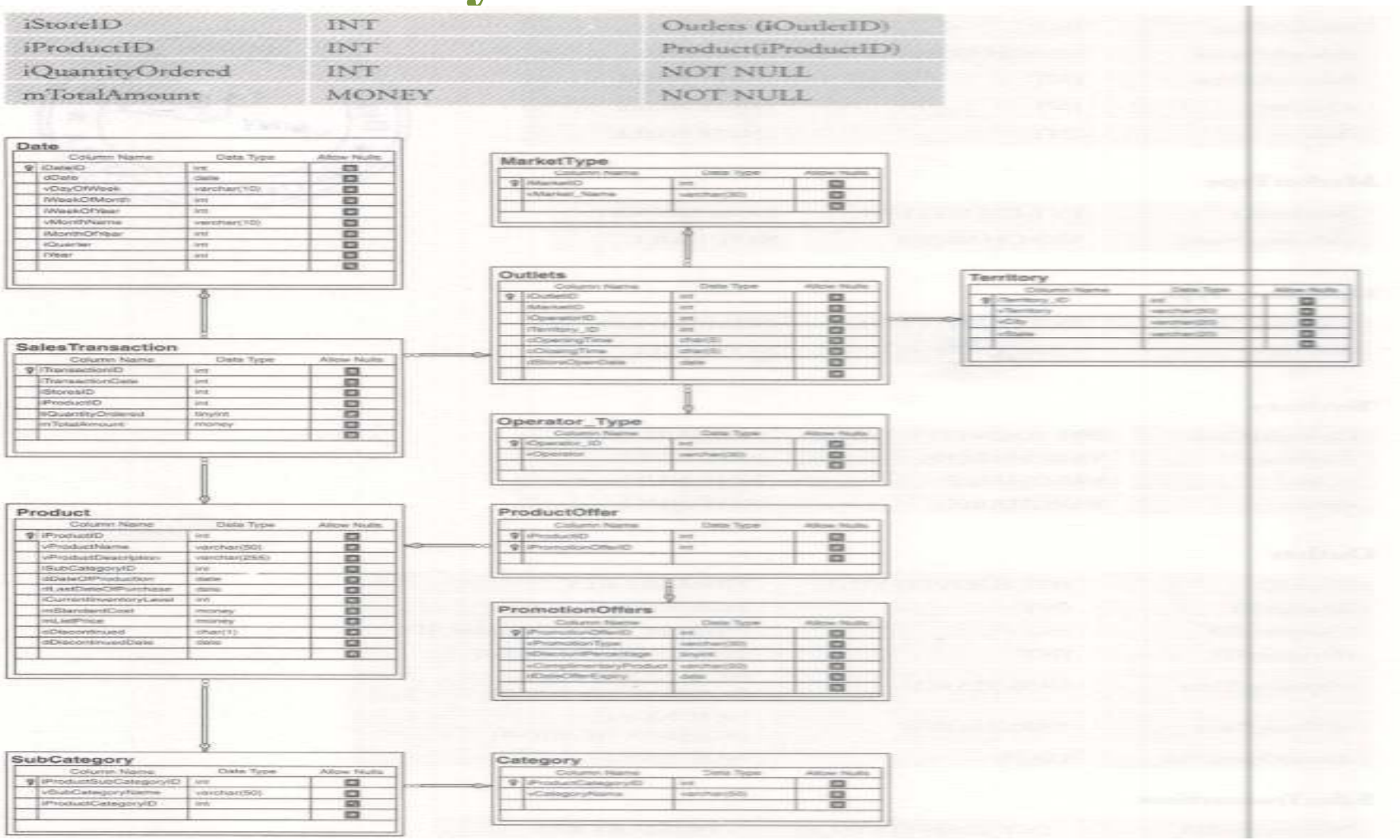
iStoreID	INT IDENTITY(1,1)	PRIMARY KEY
iMarketID	INT	Market(iMarketID)
iOperatorID	INT	Operator_Type(iOperator_ID)
iTerritoryID	INT	Territory(iTerritory_ID)
vOpeningTime	VARCHAR(5)	NOT NULL, [0-2][0-9][:][0-9][0-9]
vClosingTime	VARCHAR(5)	NOT NULL, [0-2][0-9][:][0-9][0-9]
dStoreOpenDate	DATE	NOT NULL

SalesTransaction

iTransactionID	INT IDENTITY(1,1)	PRIMARY KEY
iTransactionDate	INT	Date(iDateID)

Outcome of Physical Data Model

iStoreID	INT	Outlets (iOutletID)
iProductID	INT	Product(iProductID)
iQuantityOrdered	INT	NOT NULL
mTotalAmount	MONEY	NOT NULL



Few points of difference between Logical and Physical Data Model

- The entity names of the logical data model are table names in the physical data model.
- The attributes of the logical data model are column names in the physical data model.
- In the physical data model, the data type for each column is specified. However, data types differ depending on the actual database (MySQL,DB2, SQL Server 2008, Oracle etc.) being used. In a logical data model, only the attributes are identified without going into the details about the data type specifications.

Data Modeling Techniques – Normalization (Entity relationship) Modeling

An industry service provider, “InfoMechanists”, has several Business Units (BUs) such as

- Financial Services(FS)
- Insurance Services (IS)
- Life Science Services (LSS)
- Communication Services (CS)
- Testing Services (TS) etc.

Each BU has

- a Head as a manager
- Many employees reporting to him. Each employee has a current residential address.

Data Modeling Techniques – Normalization (Entity relationship) Modeling

- There are cases where a couple (both husband and wife) are employed either in the same BU or a different one.
- In such a case, they (the couple) have same address.
- An employee can be on a project, but at any given point in time, he or she can be working on a single project only.
- Each project belongs to a client. There could be chances where a client has awarded more than one project to the company (either to the same BU or different BUs).
- A project can also be split into modules which can be distributed to BUs according to their field of specifications.
- For example, in an insurance project, the development and maintenance work is with Insurance Services (IS) and the testing task is with Testing Services (TS). Each BU usually works on several projects at a time.

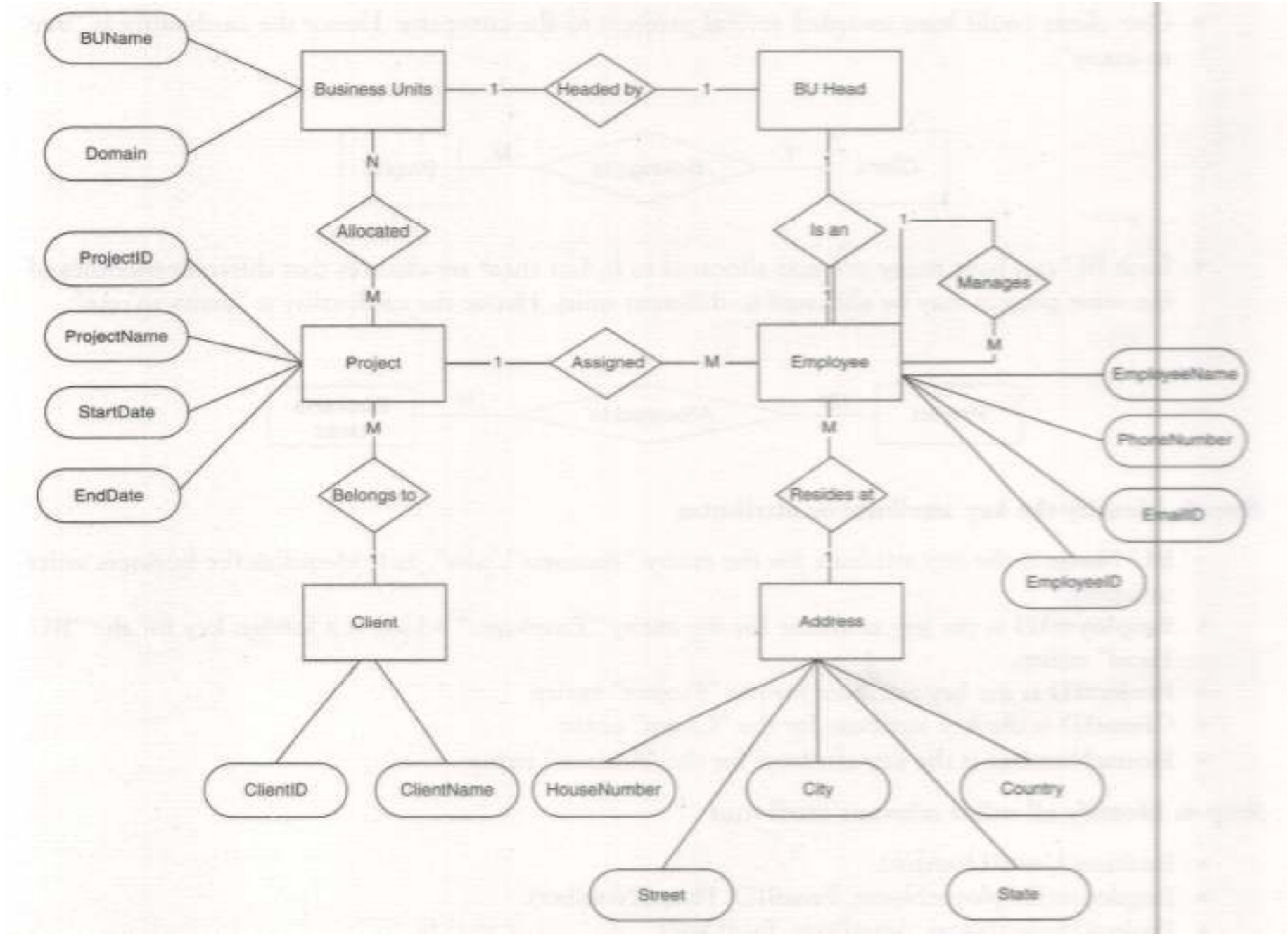
Data Modeling Techniques – Normalization (Entity relationship) Modeling

Given the specifications mentioned in the last slide, let us see how we will proceed to design an ER model.

- Enumerated below is a list of steps to help you arrive at the ER diagram:

1. Identify all the entities.
2. Identify the relationships among the entities along with cardinality and participation type(total/partial participation).
3. Identify the key attribute or attributes.
4. Identify all other relevant attributes.
5. Plot the ER diagram with all attributes including key attribute(s).
6. The ER diagram is then reviewed with the business users.

E R Model for Infomechanists



Data Modeling Techniques – Normalization (Entity relationship) Modeling Pros and Cons

Pros -

- The ER diagram is easy to understand and is represented in a language that the business users can understand.
- It can also be easily understood by a non-technical domain expert.
- It is intuitive and helps in the implementation on the chosen database platform.
- It helps in understanding the system at a higher level.

Cons –

- The physical designs derived using ER model may have some amount of redundancy.
- There is scope for misinterpretations because of the limited information available diagram.

Data Modeling Techniques – Dimensional Modeling :

Need of Dimensional Modeling

Picture this situation –

Consider you have just reached the Bangalore International Airport.

- You are an Indian national due to fly to London, Heathrow International Airport. You have collected your boarding pass.
- You have two bags that you would like checked in. The person at the counter asks for your boarding pass, weighs the bags, pastes the label with details about your flight number, your name, your travel date, source airport code, and destination airport code, etc.
- He then pastes a similar label at the back of your boarding pass. This done, you proceed to the Immigration counter, passport, and boarding pass in hand. The seal with the current date on your passport.
- Your next stop is the security counter. The security personnel scrutinize your boarding pass, passport, etc. And you find yourself in the queue to board the airfact.
- Again quick, careful rounds of verification by the aircraft crew before you find yourself ensconced in your seat.

Data Modeling Techniques – Dimensional Modeling :
Need of Dimensional Modeling

Picture this situation –

You must be wondering what has all this got to do with multidimensional modelling.

Well, we are trying to understand multidimensional perspectives of the same data. The data here is our "boarding pass".

Your boarding pass is looked at by different personnel for different reasons:

- The person at the check-in counter needs your boarding pass to book your check-in.
- The immigration personnel looked at your boarding pass to ascertain the source and of your itinerary.
- The security personnel scrutinized your boarding pass for security reasons to verify an eligible traveller.
- The aircraft crew looked at your boarding pass to onboard you and guide you to your seat.

This is nothing- but multidimensional perspectives of the same data.

To put it simply, "Multiple Perspectives".

To help with this multidimensional view of the data, we rely on dimensional modeling.

Data Modeling Techniques – Dimensional Modeling :
Need of Dimensional Modeling

Consider another Scenario-

An electronic gadget distributor company "ElectronicsForAll" is based out of Delhi, India. The company sells its products in north, north-west, and western regions of India. They have sales units at Mumbai, Pune, Ahmedabad, Delhi, and Punjab. The President of the company wants the latest sales information to measure the sales performance and to take corrective actions if required. He has requested this information from his business analysts.

Sales Report of "ElectronicsForAll" –

Representation 1 : The number of units sold = 113

Representation 2:

January	February	March	April
14	41	33	25

Representation 3:

Products	January	February	March	April
Digital Camera			6	17
Mobile Phones	6	16	6	8
Pen Drives	8	25	21	

Representation 4:

	Products	January	February	March	April
Mumbai	Digital Camera			3	10
	Mobile Phones	3	16	6	
	Pen Drives	4	16	6	
Pune	Digital Camera			3	7
	Mobile Phones	3			8
	Pen Drives	4	9	15	

Data Modeling Techniques – Dimensional Modeling- Definition :

- This method of analyzing a performance measure (in this case the number of units sold) by looking at it through various perspectives. Or in other words, the contextualized representation of a business performance measure, is known as **dimensional modeling.**
- Dimensional modeling is a logical design technique for structuring data so that it is intuitive to business users and delivers fast query performance.
- Dimensional modeling is the first step towards building a dimensional database, i.e. a data warehouse.
- It allows the database to become more understandable and simpler. In fact, the dimensional database can be viewed as a cube having three or more dimensional/perspectives for analyzing the given data.
- Dimensional modeling divides the database into two parts: (a) Measurement and (b) Context. Measurements are captured by the various business processes and other source systems.
- These measurements are usually numeric values called facts.
- Facts are enclosed by various contexts that are true at the moment the facts are recorded. These contexts are intuitively divided into independent logical clumps called dimensions. Dimensions describe the “who, what, when, where, why, and how™ context of the measurements.

To better understand the fact (measurement)—dimension (context) link, let us take the example of booking an airlines ticket. In this case, the facts and dimensions are as given below:

Facts — Number of tickets booked, amount paid, etc.

Dimensions — Customer details, airlines, time of booking, time of travel, origin city, destination city, mode of payment, etc.

Benefits of Dimensional Modeling:

1. Comprehensibility:
   - Data presented is more subjective as compared to objective nature in a relational model.
   - Data is arranged in a coherent category or dimensions to enable better comprehension.
2. Improved query performance:
3. Trended for data analysis scenarios.

Data Modeling Techniques – Dimensional Modeling- Fact Table:

Fact Table: A fact table consists of various measurements. It stores the measures of business processes and points to the lowest detail level of each dimension table. The measures are factual or quantitative in representation and are generally numeric in nature. They represent the *how much* or *how many* aspects of a question. For example, price, product sales, product inventory, etc.

Types of Fact:

Additive facts: These are the facts that can be summed up/aggregated across all dimensions in a fact table. For example, discrete numerical measures of activity — quantity sold, dollars sold, etc.

Consider a scenario where a retail store "Northwind Traders" wants to analyze the revenue generated. The revenue generated can be by the employee who is selling the products; or it can be in terms of any combination of multiple dimensions. Products, time, region, and employee are the dimensions in this case.

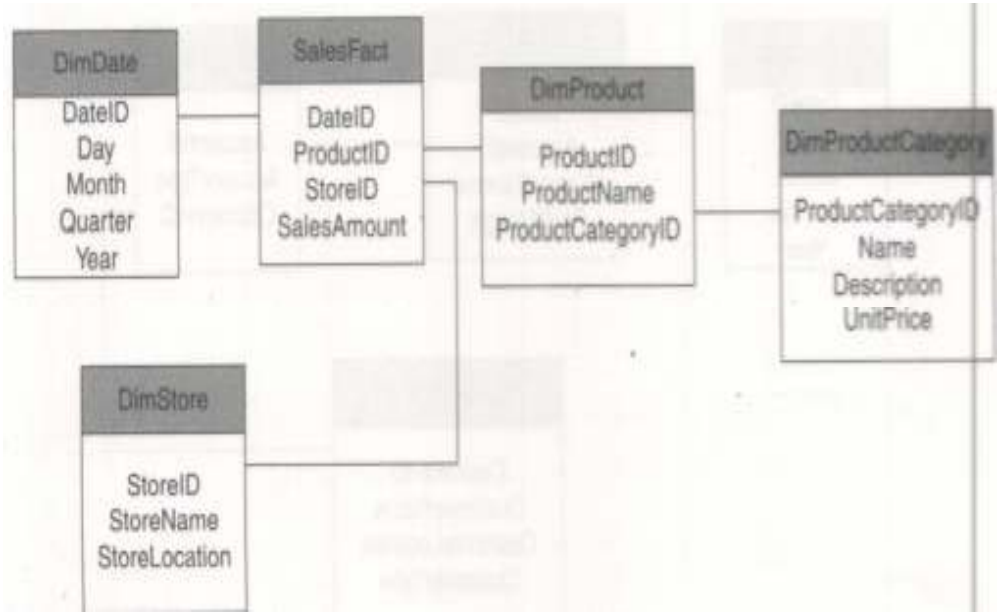
The revenue,which is a fact, can be aggregated along any of the above dimensions to give the total revenue along that dimension. Such scenarios where the fact can be aggregated along all the dimensions make the fact a fully additive or just an additive fact. Here revenue is the additive fact.

This figure depicts the "SalesFact" fact table along with its corresponding dimension tables.

This fact table has one measure., "SalesAmount", and three dimension keys, "DateID", "ProductID", and "StoreID".

The purpose of the "SalesFact" table is to record the sales amount for each product in each store on a daily basis. In this table, "SalesAmount" is an additive fact because we can sum up this fact along any of the three dimensions present in the fact table i.e. "DimDate", "DimStore", and "DimProduct".

For example – the sum of "SalesAmount" for all 7 days in a week represents the total sales amount for that week.



Data Modeling Techniques – Dimensional Modeling- Semi-Additive Facts:

Semi Additive facts: These are the facts that can be summed up for some dimensions in the fact table, but not all. For example, account balances, inventory level, distinct counts etc.

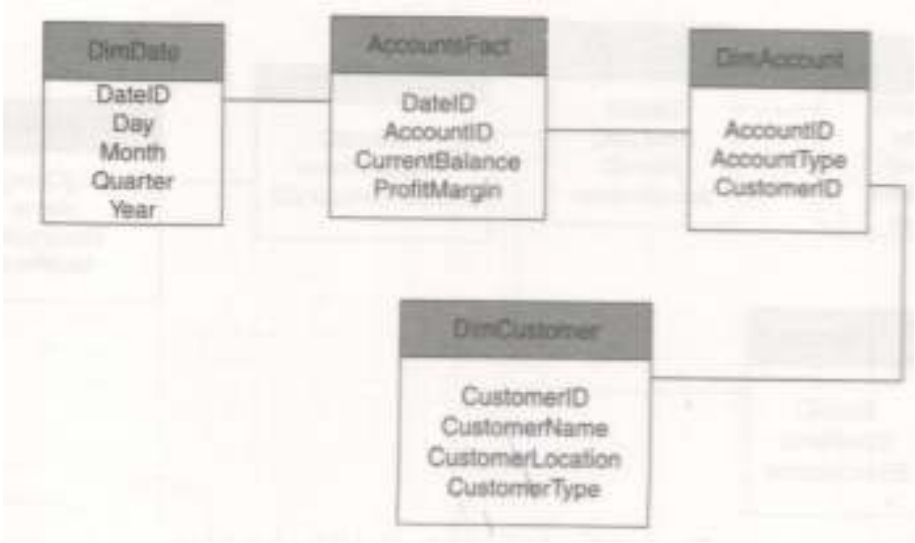
Consider a scenario where the "Northwind Traders" warehouse manager needs to find the total number of products in the inventory. One inherent characteristic of any inventory is that there will be incoming products to the inventory from the manufacturing plants and outgoing products from the inventory to the distribution centres or retail outlets.

So if the total products in the inventory need to be found out, say, at the end of a month, it cannot be a simple sum of the products in the inventory of individual days of that month. Actually, it is a combination of addition of incoming products and subtraction of outgoing ones. **This means the inventory level cannot be aggregated along the "time" dimension.**

But if a company has warehouses in multiple regions and would like to find the total products in inventory across those warehouses, a meaningful number can be arrived at by aggregating inventory levels across those warehouses. This simply means inventory levels can be aggregated along the "region" dimension. **Such scenarios where a fact can be aggregated along some dimensions but not along all dimensions give rise to semi-additive facts. In this case, the number of products in inventory or the inventory level is the semi-additive fact.**

Let us discuss another example of semi-additive facts.

Figure depicts the "AccountsFact" fact table along with its corresponding dimension tables. The "AccountsFact" fact table has two measures :"CurrentBalance" and "ProfitMargin". It has two dimension keys: "DateID" and "AccountID". "CurrentBalance" is a semi-additive fact. It makes sense to add up current balances for all accounts to get the information on "what's the total current balance for all accounts in the bank?" However, it does not make sense to add up current balances through time. It does not make sense to add up all current balances through time. It does not make sense to add up all current balance for a given account for a given account for each day of the month. Similarly, "ProfitMargin" is another non-additive fact, as it does not make sense to add profit margins at the account level or at the day level.



Data Modeling Techniques – Dimensional Modeling- Non-Additive Facts:

Non Additive facts: These are the facts that cannot be summed up for some dimensions present in the fact table. For example, measurement of room temperature, percentages, ratios, factless, facts, etc. Non additive facts cannot be added meaningfully across any dimensions. In other words, non-additive facts are facts where SUM operator cannot be used to produce any meaningful results. The following illustration will help you understand why room temperature is a non-additive fact.

Date	Temperature
5th May (7AM)	27
5th May (12 AM)	33
5th May (5 PM)	10
Sum	70 (Non-Meaningful result)
Average	23.3 (Meaningful result)

Examples of non-additive facts are:

Textual facts: Adding textual facts does not result in any number. However, counting textual facts may result in a sensible number.

Per-unit prices: Adding unit prices does not produce any meaningful number. For example: the unit sales price or unit cost is strictly non-addictive. But these prices can be multiplied with the number products sold and can be depicted as total sales amount or total product cost in the fact table.

Percentages and ratios: A ratio, such as gross margin, is non-additive. Non-additive facts are usually the result of ratio or other calculations, such as percentages.

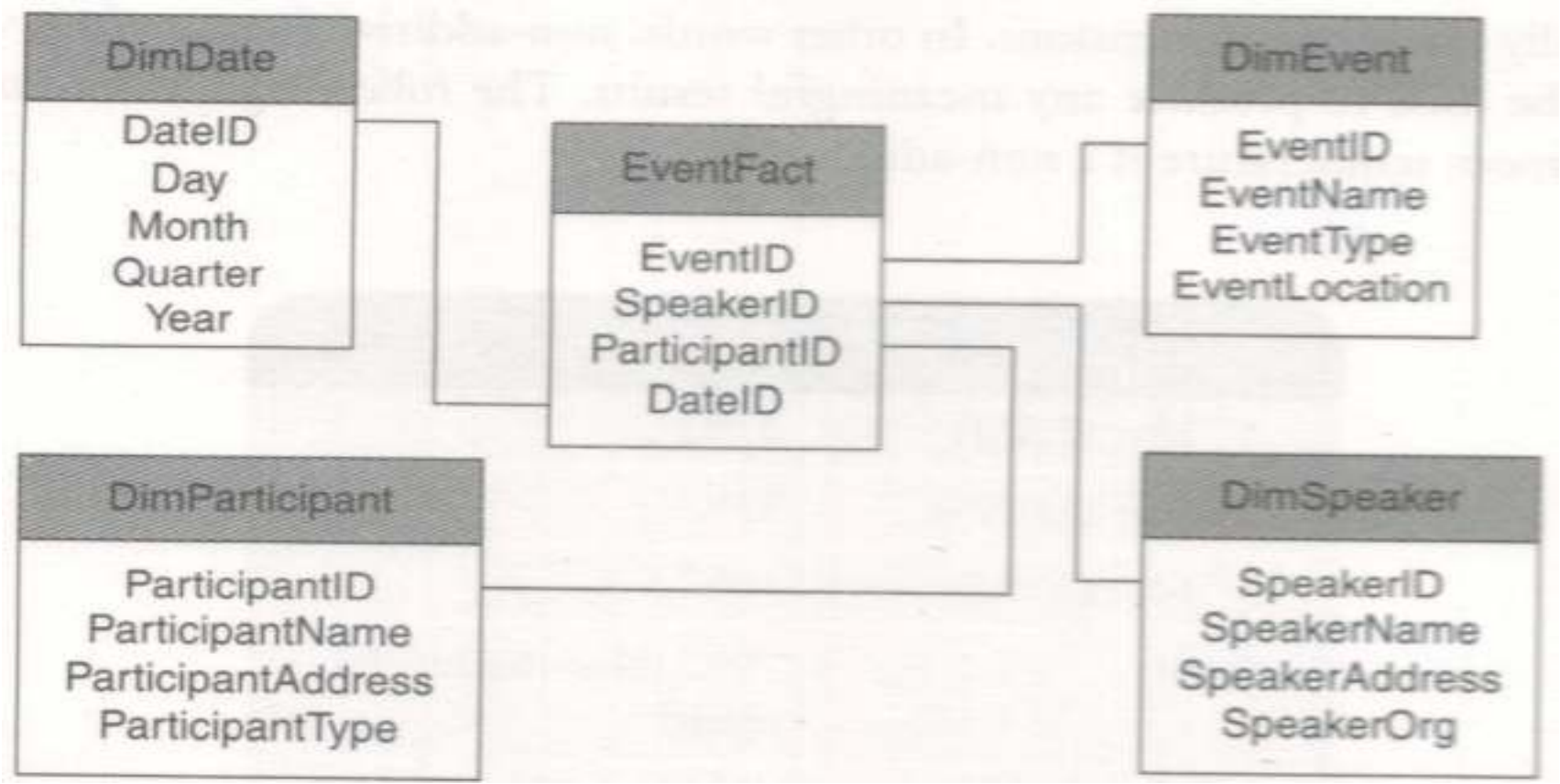
Measures of intensity: Measures of intensity such as the room temperature are non-additive across all dimensions. Summing the room temperature across different times of the day produces a totally non-meaningful number.

Averages: Facts based on averages are non-additive. For example, average sales price is non-additive. Adding all the average unit prices produces a meaningless number.

Factless facts (event-based fact tables): Event fact tables are tables that record events. For example, event fact tables are used to record events such as Webpage clicks and employee or student attendance. In an attendance recording scenario, attendance can be recorded in terms of “yes” or “no” OR with pusedo facts like “1” or “0”. In such scenarios, we can count the values but adding them will give invalid values. Factless facts are generally used to model the many-to-many relationships or to track events that did or did not happen.

Data Modeling Techniques – Dimensional Modeling- Non-Additive Facts - Example:

The following figure is an example of a “factless fact table” -“EventFact”. This factless fact table has four dimension keys: “EventID”, “SpeakerID”, “ParticipantID”, and “DateID”. It does nor have any measures or facts. This table can be queried to get details on the events that are the most popular. It can further be used to track events that did not happen. We can also use this table to elicit information about events that were the least popular or that were not attended.



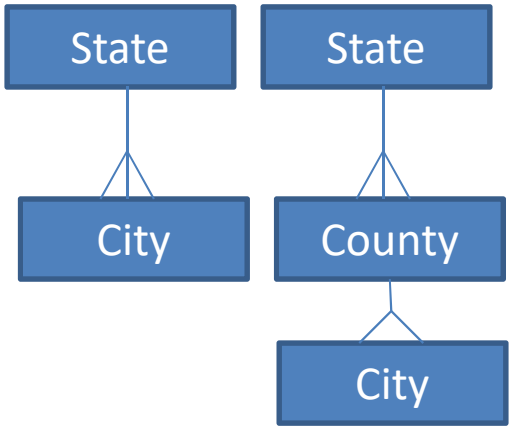
An Example of Multidimensional Modeling

Alex is excited. He will be travelling to the USA for business-related work. He has carefully planned his itinerary. Before embarking on the journey, he wants to check the weather in various US cities. He has searched the Internet to get the required information for the coming week. He has a table of data before him which looks like as shown below:

City Name	DateDetails	MinTemp	MaxTemp
Los Angels	22-05-2011	86	105
San Frasisco	22-05-2011	78	107
Phoenix	22-05-2011	88	98
Los Angels	23-05-2011	82	106
San Francisco	23-05-2011	76	104
Phoenix	23-05-2011	86	96

In the above table, we have two dimensions, say, the “Geography” dimension and the “Time” dimension. “NameofCity” and “DateDetails” are attributes of the geography and time dimension respectively. There are also two facts, “MinTemp” and “MaxTemp”. Using this table, it is possible to find out information about the maximum daily temperatures and the minimum daily temperatures for any group of cities or group of days. Now let us assume that we wish to view the maximum and minimum temperatures for states. A city belongs to a state. Let us add an attribute “State” to the “Geography” dimension. The relationship between the state and the city is as depicted in the following figure:

A state can have multiple cities. The relationship is one-to-many from the state to cities. Now assume that we wish to have a look at the minimum and maximum temperatures by counties. This can be achieved by adding yet another attribute “County” to the geography dimension. The relationship between the state and county is as depicted in figure. The relationship is many from the state to counties. You already know that temperature is a non-additive fact. However, one can look at the average temp for cities or states or for different time periods or for a combination of geography and time.



What Are Dimensions/Dimension Tables?

- Dimension tables consist of dimension attributes which describe the dimension elements to enhance comprehension.
- Dimension attributes (descriptive) are typically static values containing discrete numbers which behave as text values.
- **Main functionalities :**
  - ❑ Query filtering\constraining
  - ❑ Query result set labeling
- **The dimension attribute must be**

  Complete: Dimension attributes must not contain missing values.

  Verbose: Labels must consist of full words.

  Descriptive: The dimension attribute names must be able to convey the purpose of the dimension element in as few and simple words as possible.

  Discrete values: Dimension attributes must contain only one value per row in dimension table.

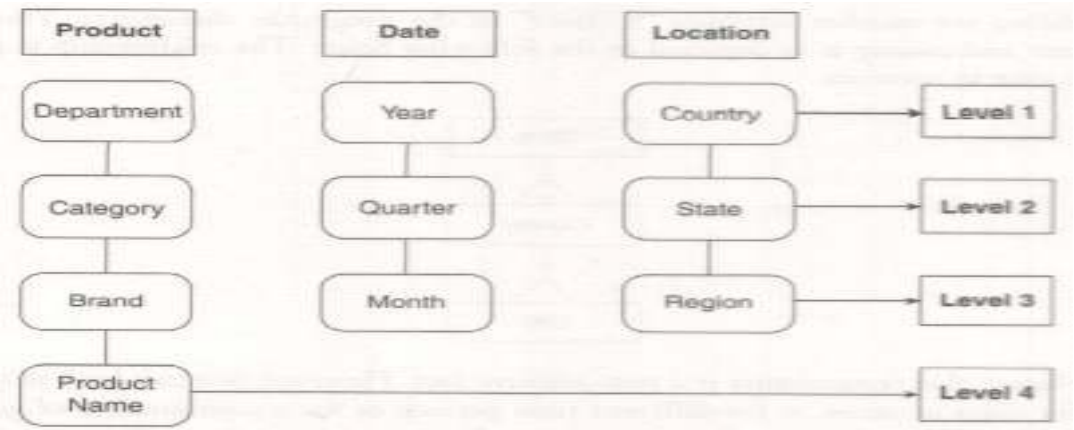
  Quality assured: Dimension attributes must not contain misspelt values or impossible values.

Dimension Hierarchies

- A dimension hierarchy is a cascaded series of many-to-one relationships and consists of different levels. Each level in a hierarchy corresponds to a dimension attribute. Hierarchies document the relationship between different levels in a dimension.
- A dimension hierarchy may also be described as a set of parent-child relationships attributes present within a dimension. These hierarchy attributes, also known as levels, roll up a child to parent. For example, Customer totals can roll up to Sub-region totals which can further roll up to Region totals. A better example would be — daily sales could roll up to weekly sales, which further roll up to month to quarter to yearly sales. Let us understand the concept of hierarchy through the example. In this example, the Product hierarchy is like this

Department ⟶ Category ⟶ Brand ⟶ Product Name

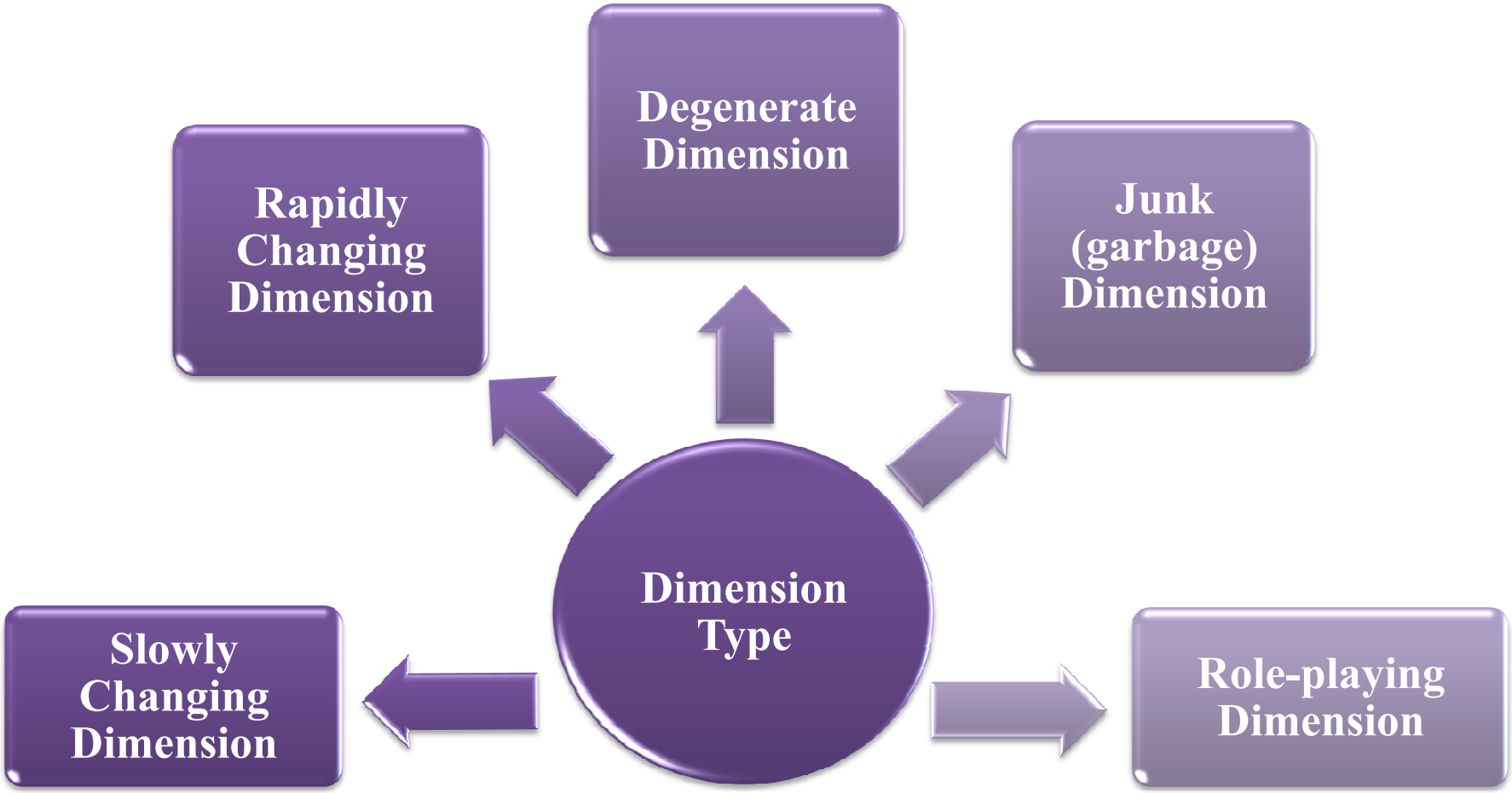
Dimension Hierarchies - Example



Similarly, the Date hierarchy is depicted as
Year ⟶ Quarter ⟶ Month
Example: 2011 ⟶ Q1 ⟶ April

For a better idea of dimension hierarchy, let us assume a product store, “ProductsForAll”. The store has several departments such as “Confectionary”, “Electronics”, “Travel Goods”, “Home Appliances”, “Dairy Products”, etc. Each department is further divided into categories. Example “Dairy Products” further classified into “Milk”, “Butter”, “Cottage Cheese”, “Yogurt”, etc. Each product class offers several brands such as “Amul”, “Nestle”, etc. And, finally each brand has specific product example, “Amul cheese” has names such as “Amul Slim Cheese”, “Amul EasySpread”, etc.

Types of Dimensions



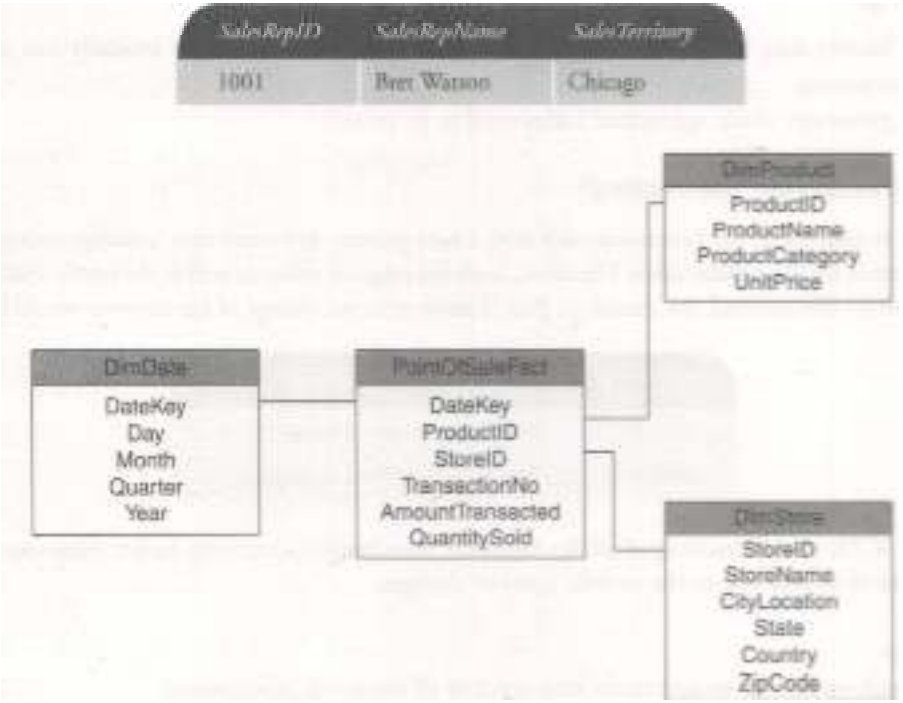
Dimension Tables – Degenerate Dimension

A degenerate dimension is a data that is dimension in temperament but is present in a fact table. It is a dimension without any attributes. Usually, a degenerate dimension is a transaction-based number. There can be more than one degenerate dimension in a fact table.

Degenerate dimensions often cause confusion as they don't feel or look like normal dimensions. They act as dimension keys in fact tables; however, they are not joined to corresponding dimensions in other dimension tables as all their attributes are already present in other dimension tables.

Degenerate dimensions can also be called textual facts, but they are not facts as the primary key for the fact table is often a combination of dimensional foreign keys and degenerate dimensions. As already stated, a fact table can have more than one degenerate dimension. For example, an insurance claim line fact table typically includes both claim and policy numbers as degenerate dimensions. A manufacturer can include degenerate dimensions for the quote, order, and bill of lading numbers in the shipments fact table.

This figure depicts a PointOfSalesFact table along with other dimension tables. The "PointOfSalesFact" has two measures: AmountTransacted and QuantitySold. It has the following dimension keys: DateKey that links the "PointOfSaleFact" to "DimDate", ProductID that links the "PointOfSaleFact" to "DimProduct" and "StoreID" that links the "PointOfSaleFact" to "DimStore". Here, TransactionNo is a degenrate dimension as it is a dimension key without a corresponding dimension table. All information/details pertaining to the transaction are extracted and stored in the "PointOfSaleFact" table itself;therefore, there is no need to have a separate dimension table to store the attributes of the transaction.



Dimension Tables – Slowly Changing Dimension (SCD)

In a dimension model, dimension attributes are not fixed as their values can change slowly over a period of time. Here comes the role of a slowly changing dimension. A slowly changing dimension is a dimension whose attribute/attributes for a record (row) change slowly over time, rather than change on a regularly timely basis. Let us assume a company sells car-related accessories. The company decides to assign a new sales territory, Los Angeles, to its sales representative, Bret Watson, who earlier operated from Chicago. How can you record the change without making it appear that Watson earlier held Chicago?

Let us take a look at the original record of Bret Watson: Now the original record has to be changed as Bret Watson has been assigned "Los Angeles" as his sales territory, effective May 1, 2011. This would be done through a slowly changing dimension. Given below are the approaches for handling a slowly changing dimension:

Type-I (Overwriting the History)

In this approach, the existing dimension attribute is overwritten with new data, and hence no history is preserved. This approach is used when correcting data errors present in a field, such as a word spelled incorrectly.

SalesRepID	SalesRepName	SalesTerritory
1001	Bret Watson	LosAngels

Type-II (Preserving the History)

A new row is added into the dimension table with a new primary key every time a change occurs to any of the attributes in the dimension table. Therefore, both the original values as well as the newly updated values are captured.

SalesRepID	SalesRepName	SalesTerritory
1001	Bret Watson	Chicago
1006	Bret Watson	Los Angeles

Type-III(Preserving One or more Versions of History)

This approach is used when it is compulsory for the data warehouse to track historical and when these changes will happen only for a finite number of times. Type-III SCDs do not increase the size of the table as compared to the Type-II SCDs since old information is updated by adding new information.

SalesRepID	SalesRepName	OriginalSalesTerritory	CurrentSalesTerritory	EffectiveFrom
1001	Bret Watson	Chicago	Los Angeles	01-05-2011

Dimension Tables – Slowly Changing Dimension (SCD)

Type-I (Overwriting the History)

Advantages

•It is the easiest and simplest approach to implement.
•It is very effective in those situations requiring the correction of bad data.
•No change is needed to the structure of the dimension table.

Disadvantages

•All history may be lost in this approach if used inappropriately.
•It is typically not possible to trace history.
•All previously made aggregated tables need to be rebuilt.

Type-II (Preserving the History)

Advantages

•This approach enables us to accurately keep track of all historical information.

Disadvantages

•This approach will cause the size of the table to grow fast.
•Storage and performance can become a serious concern, especially in cases where the number of rows for the table is very high to start with.
•It complicates the ETL process too.

Type-III(Preserving One or more Versions of History)

Advantages

Since only old information is updated with new information, this does not increase the size of the table.
It allows us to keep some part of history.

Disadvantages

Type-III SCDs will not be able to keep all history where an attribute is changed more than once.
For example, if Bret Watson is later assigned “Washington” on December 1, 2012, the Los Angeles information will be lost.

Dimension Tables – Slowly Changing Dimension (SCD)-

Comparison of the three types of handling of SCD

	Type-I	*Type-II*	*Type-III*
When to use	• When the attribute change is simple. • Tracking of history not required.	• To keep a track of all the historical changes.	• To keep a track of a finite number of historical changes.
Advantage	• Easiest and simplest to implement. • Effective in performing data correction. • No change in data structure.	• Enables tracking of historical changes accurately. • Can track infinite number of changes.	• Does not increase the size as Type-II, as old information is updated. • Keeps a part of the history, equivalent to the number of changes predictable.
Disadvantage	• All history is lost if used inappropriately. • Previous aggregated tables have to be remade.	• Dimension table grows fast. • Complicated ETL process to load the dimension model.	• No complete history, especially when change occurs very often. • Risk of losing history or changing design if more history has to be traced.
Impact on existing dimension tables	• No impact. No change in table structure.	• No impact. No change in table structure.	• Dimension table is modified to accommodate additional columns. • Number of columns based on number of changes to track.
Impact on pre-aggregation	• All pre-existing aggregations have to be re-built.	• No impact. Aggregate tables need not be re-built.	• All pre-existing aggregations have to be re-built.
Impact on database size	• No impact on the size of the database.	• Accelerated growth as a new row is added every time a change occurs.	• No impact as the data is only updated.

Dimension Tables – Rapidly Changing Dimension (RCD)

We have seen how to handle very slow changes in the dimension, but what would happen if occur more frequently?

A dimension is considered to be a fast changing dimension, also call changing dimension, if its one or more attributes change frequently and also in several rows. For example, consider a customer table having 1,00,000 rows. Assuming that on an average 10 changes occur in a dimension every year, then in one year the number of rows will increase to 1,00,000 x 10 = 10,00,000.

To identify a fast changing dimension, look for attributes having continuously variable values. Some of the fast changing dimension attributes have been identified as:

- Age
- Income
- Test score
- Rating
- Credit history score
- Customer account status
- Weight

One method of handling fast changing dimensions is to break off a fast changing dimension into one or more separate dimensions known as mini-dimensions. The fact table would then have two separate foreign keys — one for the primary dimension table and another for the fast changing attribute.

Dimension Tables – Junk Garbage Dimension (JGD)

The garbage dimension is a dimension that contains low-cardinality columns/attributes such as indicators, codes, and status flags. The garbage dimension is also known as junk dimension. The attributes in a garbage dimension are not associated with any hierarchy.

We recommend going for junk/ garbage dimension only if the cardinality of each attribute is relatively low, there are only a few attributes, and the cross-join of the source tables is too big. The option here will be to create a junk dimension based on the actual attribute combinations found in the source data for the fact table. This resulting junk dimension will include only combinations that actually occur, thereby keeping the size significantly smaller.

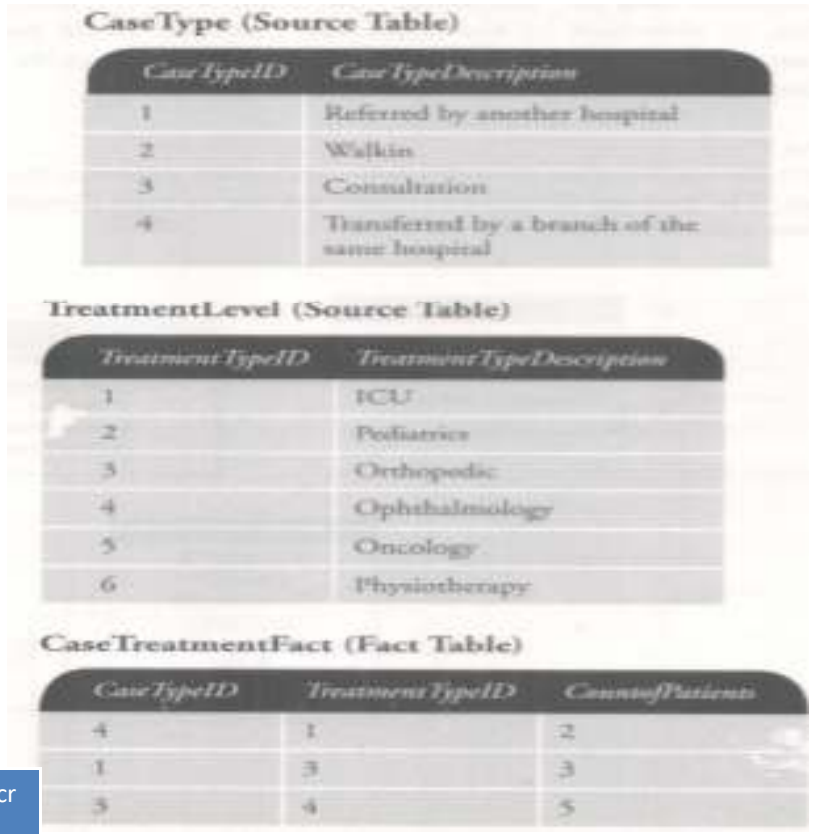
A junk dimension will combine several low cardinality flags and attributes into a single table rather than modeling them as separate dimensions. This will help reduce the size of the fact table and make dimensional modeling easier to work with.

Let us look at the following example from the healthcare domain. There are two source tables and a fact table:

In our example, each of the source tables
[CaseType (Case-TypeID, CaseTypeDescription) and TreatmentLevel (Treatment TypeID, Treatment TypeDescription)]
has only two attributes each. The cardinality of each attribute is also low.

One way to build the junk dimension will be to perform a cross-join of the source tables. This will create all possible combinations of attributes, even if they do not or might never exist in the real world.

The other way is to build the junk dimension based on the actual attribute combinations found in the source tables for the fact table. This will most definitely keep the junk dimension table significantly smaller since it will include only those combinations that actually occur. Based on this explanation, we redesign the fact table along with the junk dimension table as shown below:



CaseType (Source Table)

CaseTypeID	CaseTypeDescription
1	Referred by another hospital
2	Walkin
3	Consultation
4	Transferred by a branch of the same hospital

TreatmentLevel (Source Table)

TreatmentTypeID	TreatmentTypeDescription
1	ICU
2	Pediatrics
3	Orthopedic
4	Ophthalmology
5	Oncology
6	Physiotherapy

CaseTreatmentFact (Fact Table)

CaseTypeID	TreatmentTypeID	CountofPatients
4	1	2
1	3	3
3	4	5

SurrogateKeyID	CountOfPatients
1	2
2	3
3	5

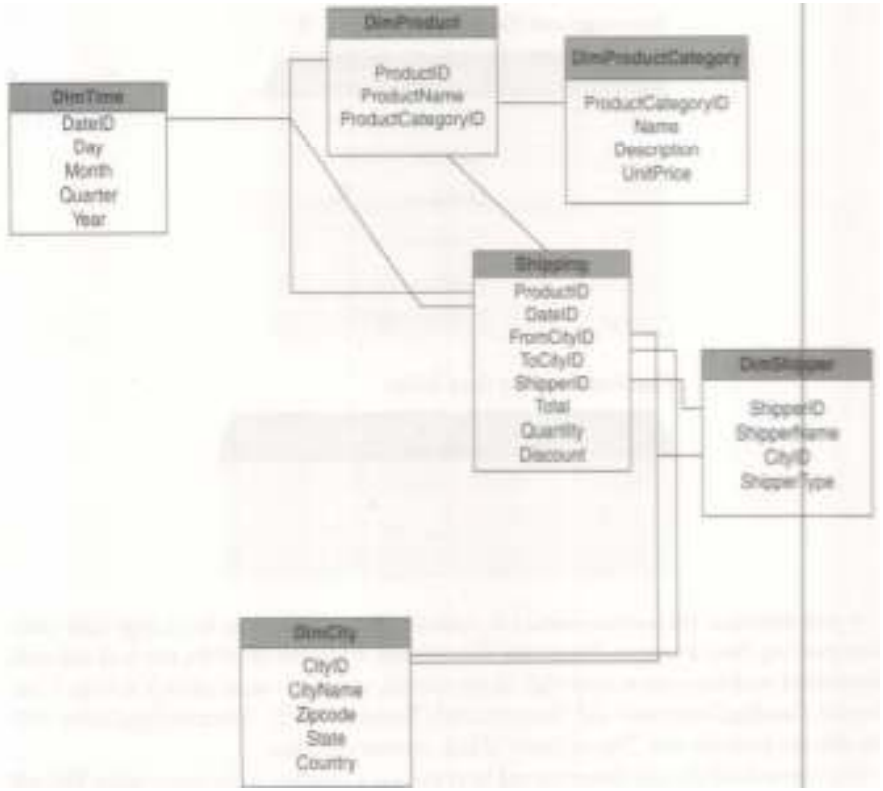
SurrogateKeyID	CaseTypeID	CaseTypeDescription	TreatmentTypeID	TreatmentTypeDescription
1	4	Tsrf by a brnch	1	ICU
2	1	Rfrd by anthr hsp	3	Orthopaedic
3	3	Consultaion	4	Ophthalmology

Dimension Tables – Role Playing Dimension (RPD)

A single dimension that is expressed differently in a fact table with the usage of views is called a role-playing dimension.

Consider an on-line transaction involving the purchase of a laptop. The moment an order is placed, an order date and a delivery date will be generated. It should be observed that both the dates are the attributes of the same time dimension. Whenever two separate analyses of the salts performance — one in terms of the order date and the other in terms of the delivery date — are required, two views of the same time dimension will be created to perform the analyses. In this scenario, the time dimension is called the role-playing dimension as it is playing the role of both the order and delivery dates.

Another example of the role-playing dimension is the broker dimension. The broker can play the role of both sell broker and buy broker in a share trading scenario. Figure below will help you a better understanding of the role-playing dimension.



"Shipping" is a fact table with three measures — "Total", "Quantity", and "Discount.

It has five dimension keys —

"ProductID" that links the fact table "Shipping" with the "DimProduct" dimension table;

"DateID" that links "Shipping" with the "DimTime" dimension table;

"ShipperID" that links "Shipping" with the "DimShipper" dimension table;

and the remaining two dimensions "ToCityID" and "FromCityID", link the "Shipping" fact table with the same dimension table, i.e. "DimCity".

The two cities, as identified by the respective CityIDs, would have the same(DimCity) but would mean two completely different cities when used to signify FromCity and ToCity. This is a case of role-playing dimension.

Typical Dimension Models

As it has been earlier discusses that the Entity Relationship (ER) data model is a commonly used data model for relational databases. Here, the database schema is represented by a set of entities and the relationship between them. It is an ideal data model for On-Line Transaction Processing (OLTP).
Let us look at a data model that is considered apt for On-Line Data Analysis. Multidimensional data modeling is the most popular data model when it comes to designing a data warehouse.

Dimensional modeling is generally represented by either of the following schemas

1. Star Schema
2. Snowflake Schema
3. Fact Constellation Schema

Typical Dimension Models – Star Schema

- It is the simplest of data warehousing schema.
- It consists of a large central table (called the fact table) with no redundancy.
- The central table is being referred by a number of dimension tables. The schema graph looks like a starburst (see figure below).
- The dimension tables form a radial pattern around the large central fact table.
- The star schema is always very effective for handling queries.
- In the star schema, the fact table is usually in 3NF or higher form of normalization.
- All the dimension tables are usually in a denormalized manner, and the highest form of normalization they are usually present in is 2NF.
- The dimension tables are also known as look up or reference tables.

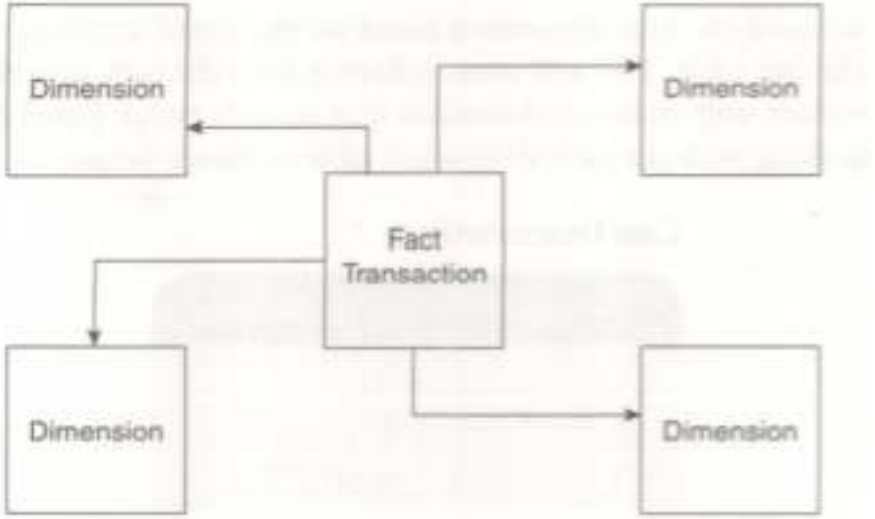


Figure 7.11 The data model for star schema.

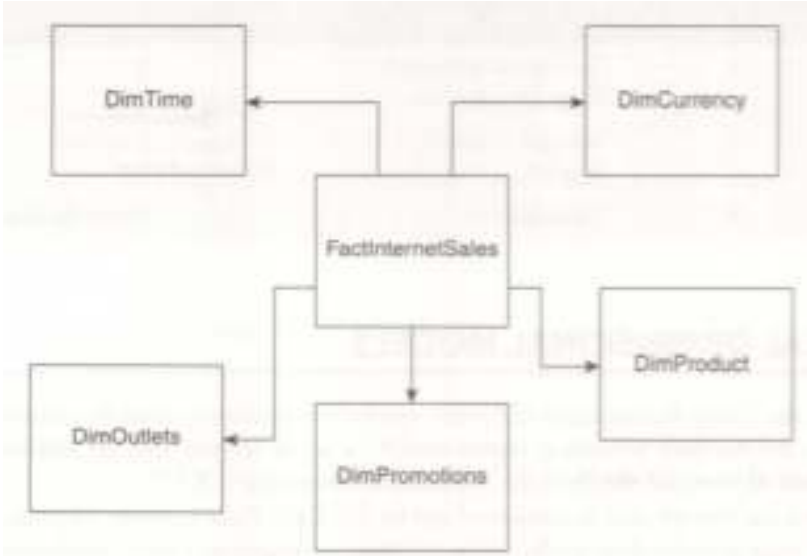
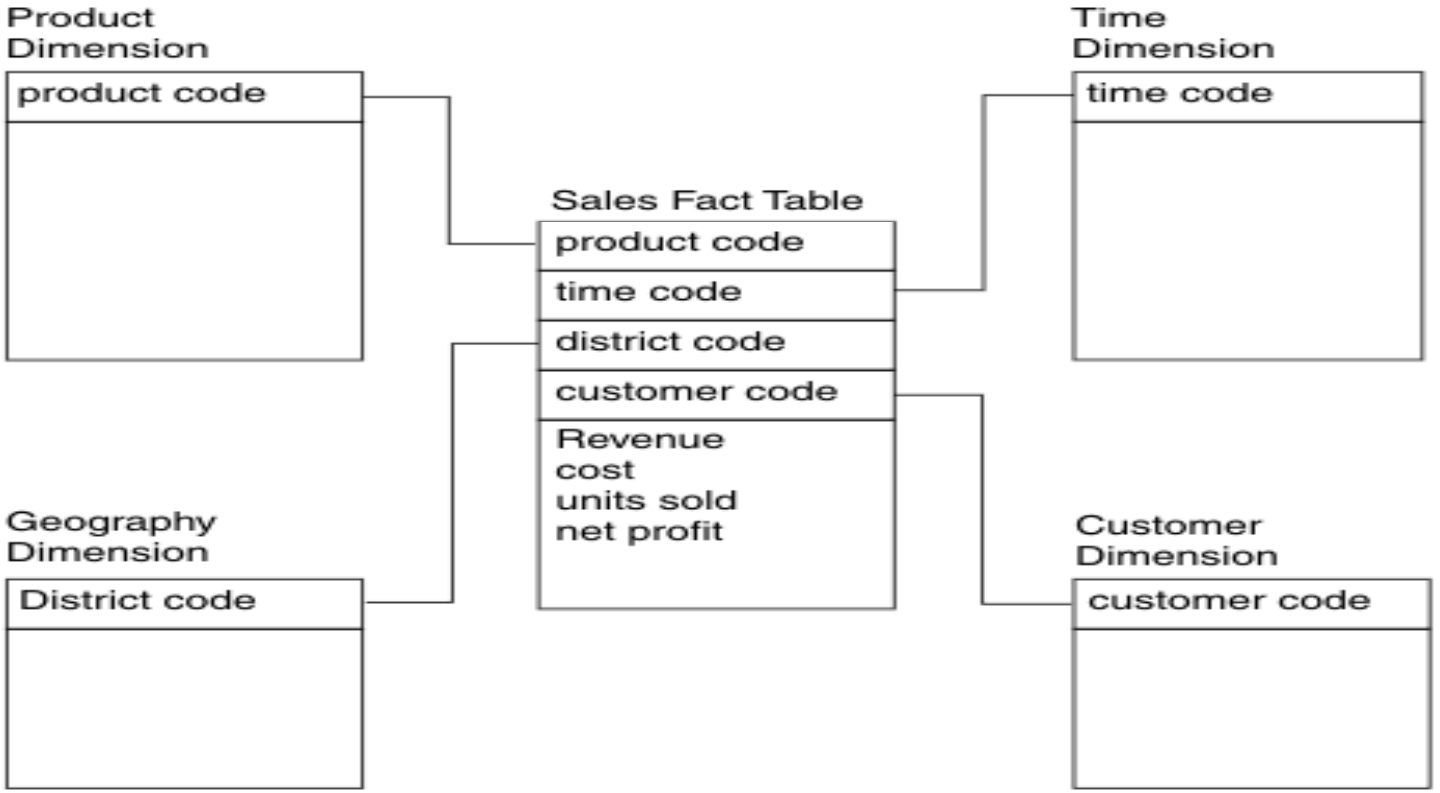


Figure 7.12 The data model for "TenToTen" Stores in Star Schema.

Example - Star Schema for sales of “ElectronicsForAll”

The basic star schema contains four components.
These are:

Fact table, Dimension tables, Attributes and Dimension hierarchies



Snow Flake Schema

- The Snowflake schema is a variant of the Star schema.
- Here, the centralized fact table is connected to multiple dimensions.
- In the Snowflake schema, dimensions are present in a normalized form in multiple related tables (Figure below).
- A snowflake structure materializes when the dimensions of a star schema are detailed and highly structured, having several levels of relationship, and the child tables have multiple parent tables.
- This "snowflaking" effect affects only the dimension tables and does not affect the fact table.

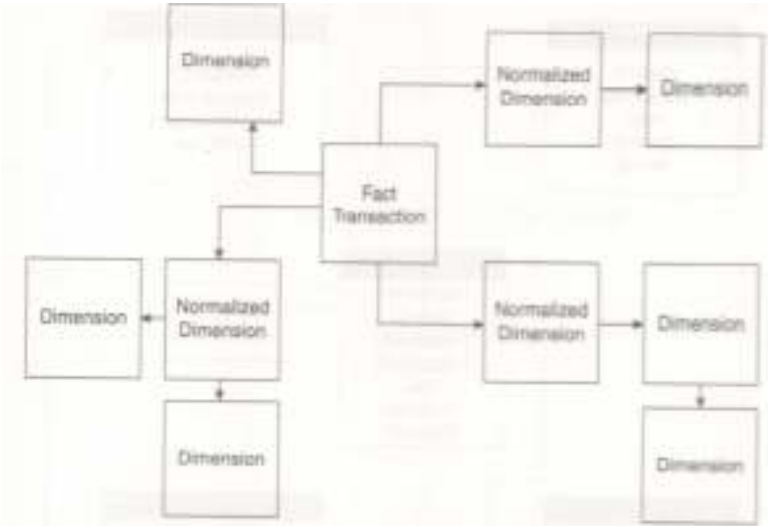


Figure 7.14 The data model for Snowflake schema.

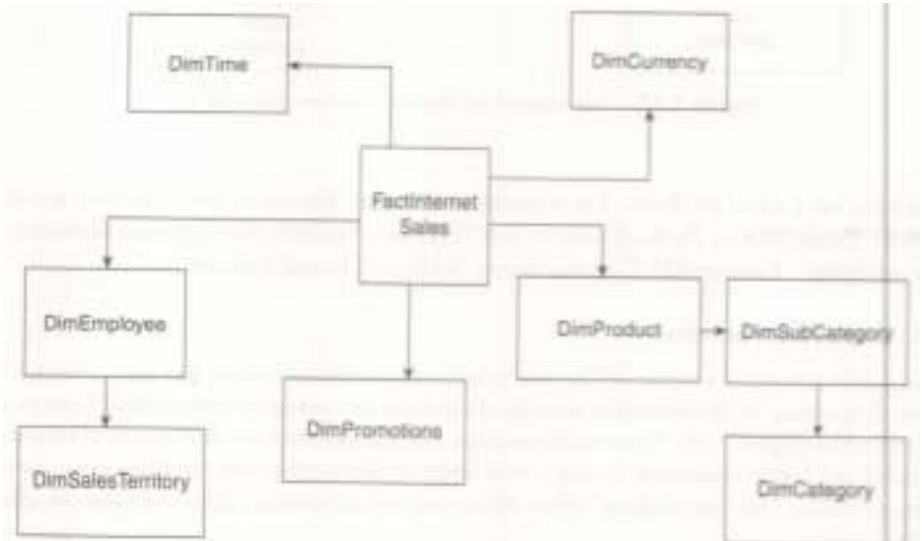


Figure 7.15 The data model for "TenToTen" Stores in Snowflake schema.

Snow Flake Schema

- Normalization and expansion of the dimension tables in a star schema result in the implementation of a snowflake design.
- A dimension table is said to be snow flaked when the low-cardinality attributes in the dimension have been removed to separate normalized tables and these normalized tables are then joined back into the original dimension table.

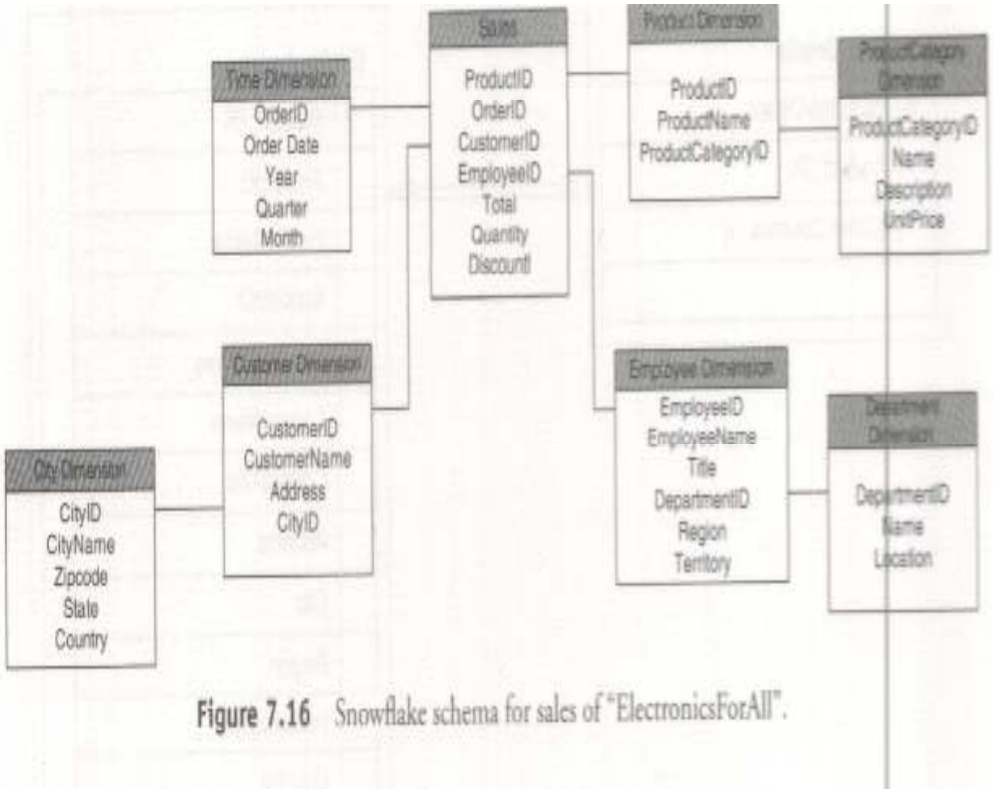


Figure 7.16 Snowflake schema for sales of "ElectronicsForAll".

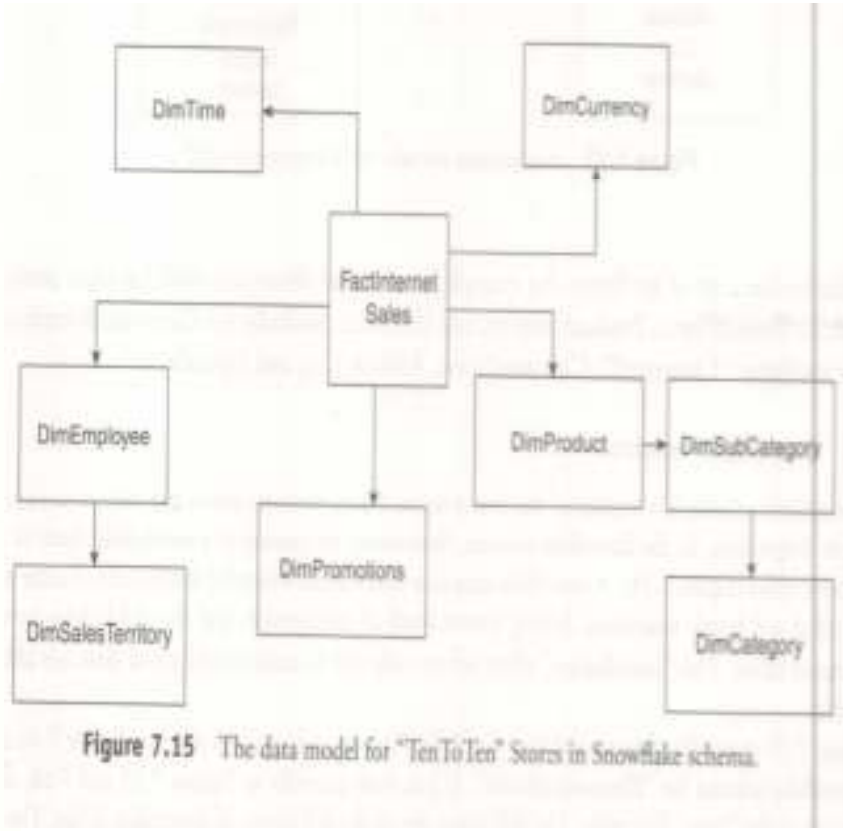


Figure 7.15 The data model for "TenToTen" Stores in Snowflake schema.

Snow Flake Schema

As we have in the example of “ElectronicsForAll”, the main difference between the Star and Snowflake schema is that the dimension tables of the Snowflake schema are maintained in normalized form to reduce redundancy. The advantage here is that such tables (normalized) are easy to save storage space. However, it also means that more joins will be needed to execute a query. This will adversely impact system performance.

Identifying Dimensions to be Snowflaked

In this section, we will observe the practical implementation of the dimensional design.

What is snowflaking?

The snowflake design is the result of further expansion and normalization of the dimension table. In other words, a dimension table is said to be snowflaked if the low-cardinality attributes of the dimensions have been divided into separate normalized tables. These tables are then joined to dimension table with referential constraints (foreign key constraints).

Generally, snowflaking is not recommended in the dimension table, as it hampers the understandability and performance of the dimensional model as more tables would be required to satisfy the queries.

When do we snowflake?

The dimensional model is snowflaked under the following two conditions:

The dimension table consists of two or more sets of attributes which define information at different grains.The sets of attributes of the same dimension table are being populated by different source systems.

Snow Flake Schema

For understanding why and when we snowflake, consider the “Product” dimension tabl¢ shown in

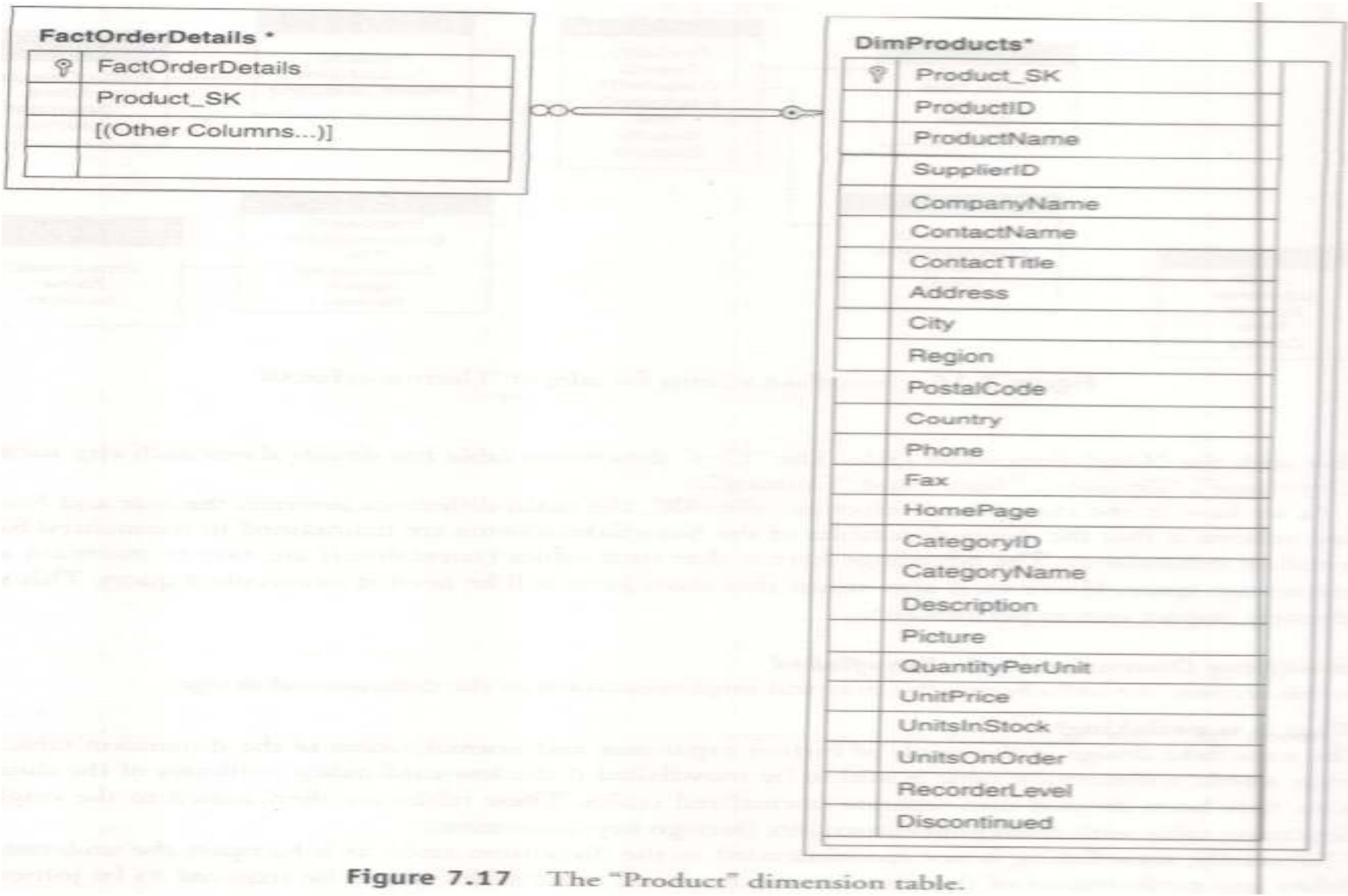


Figure 7.17 The “Product” dimension table.

Conversion to Snowflaked Schema

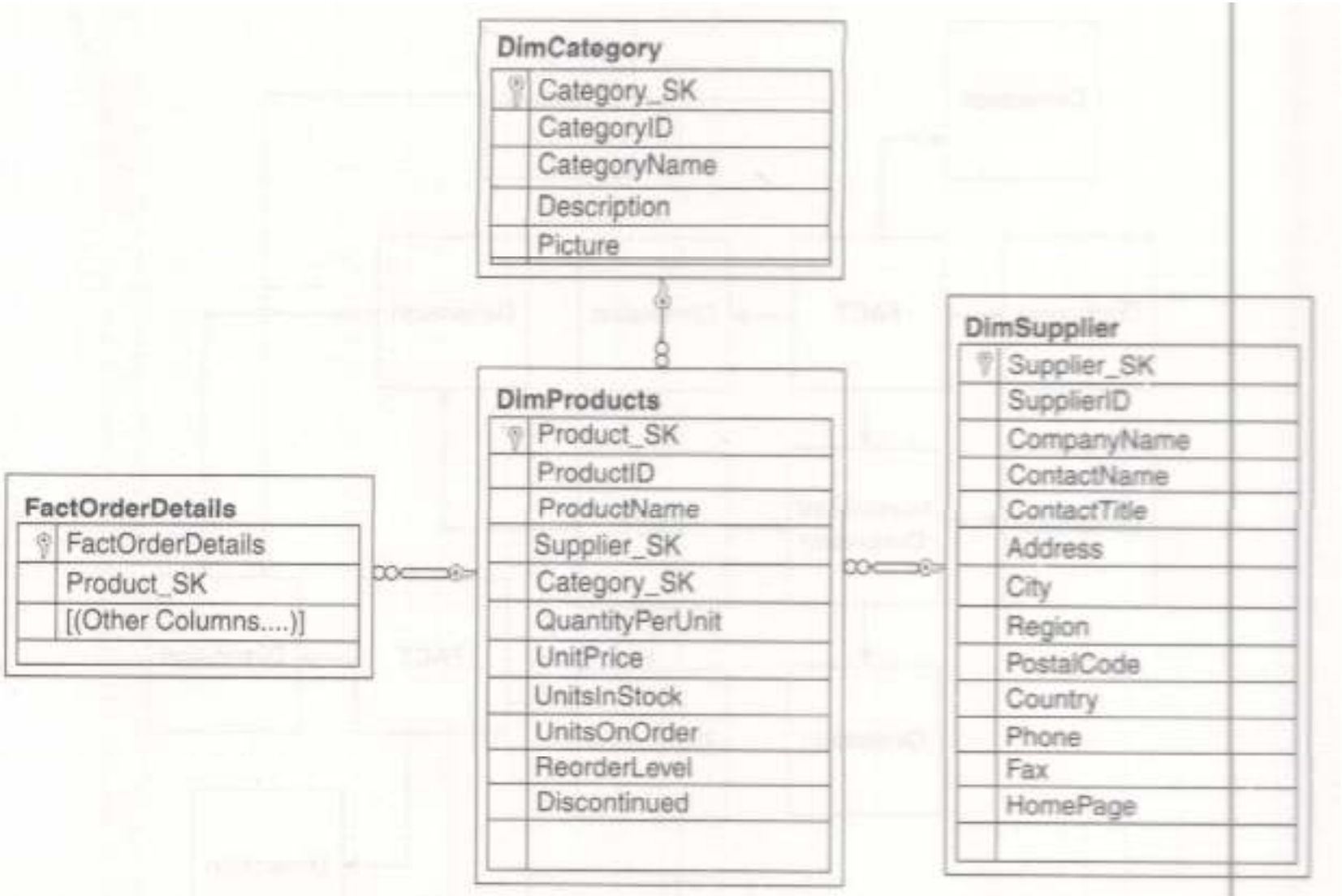
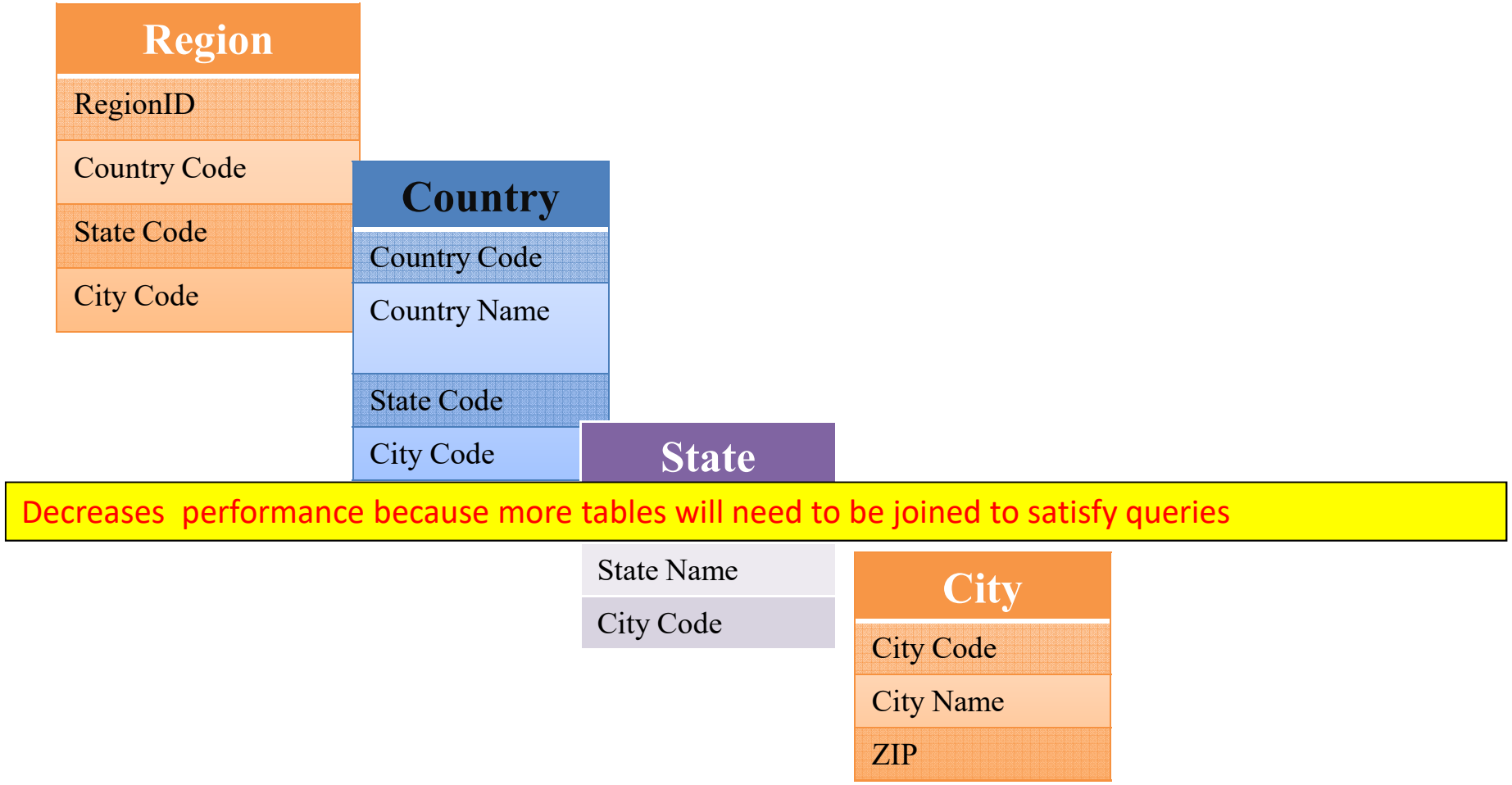


Figure 7.18 The data model for the "Product" table in Snowflaked schema.

Snow Flaking Example

- Consider the Normalized form of Region dimension



Why not to Snowflake?

Normally, you should avoid snowflaking or normalization of a dimension table, unless required and appropriate. Snowflaking reduces space consumed by dimension tables, but compared with entire data warehouse the saving is usually insignificant.

Do not snowflake hierarchies of one dimension table into separate tables. Hierarchies should belong to the dimension table only and should never be snowflaked. Multiple hierarchies can belong to the same dimension if the dimension has been designed at the lowest possible detail.

Data Model for Fact Constellation Schemaof TenToTen Stores

The constellation schema is shaped like a constellation of stars (i.e. Star schemas). This is more complex than Star or Snowflake schema variations, as it contains multiple fact tables. This allows the dimension tables to be shared among the various fact tables. It is also called “Galaxy schema”. The main disadvantage of the fact constellation is more complicated design because multiple aggregations must be taken into consideration (Figure below).

Figure 7.19 The data model for Fact Constellation schema.

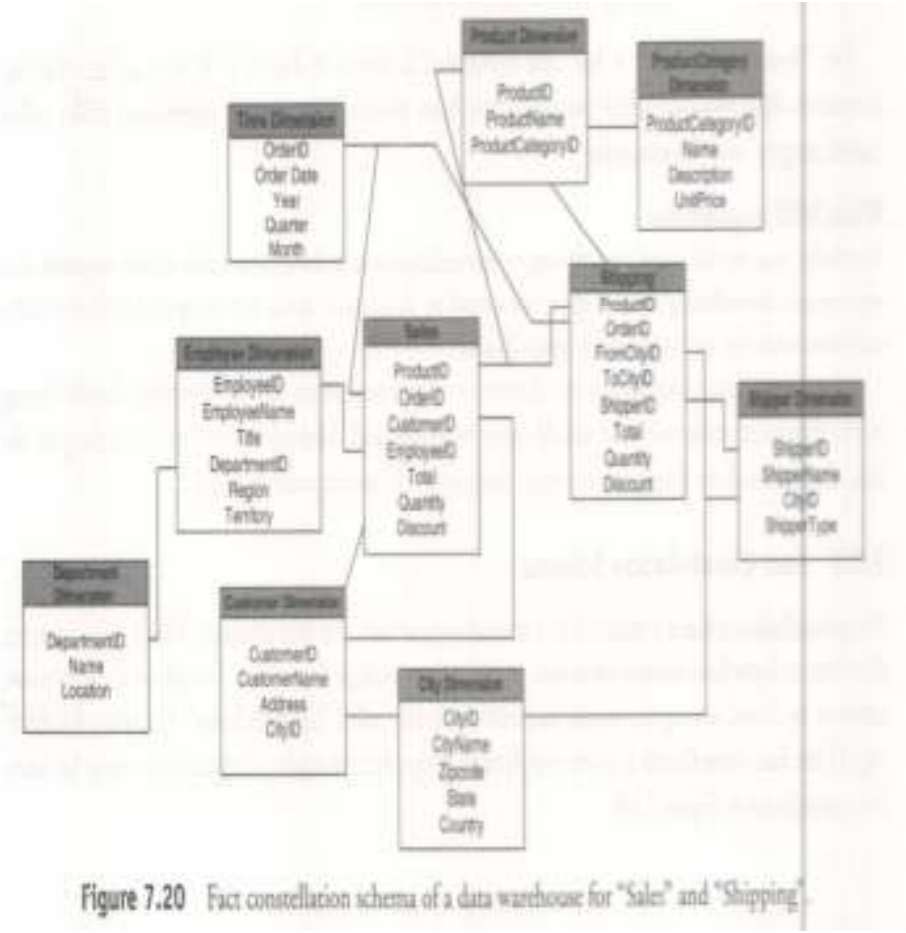


Figure 7.20 Fact constellation schema of a data warehouse for “Sales” and “Shipping”.

Dimensional Modeling Life Cycle

Phases of Dimensional Modeling Life Cycle:

1. Requirements gathering
2. Identifying the grain
3. Identifying the dimensions
4. Identifying the facts
5. Designing the dimensional model

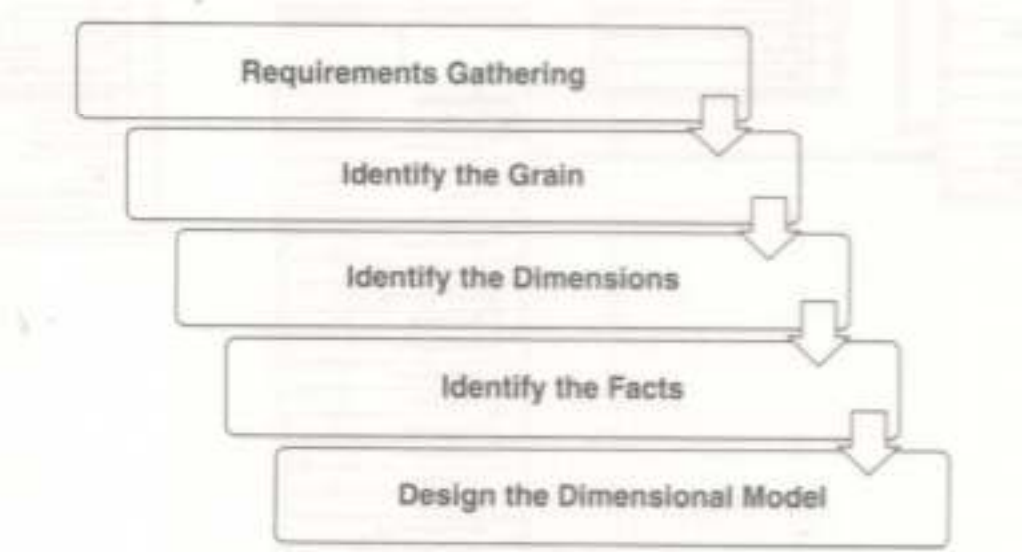
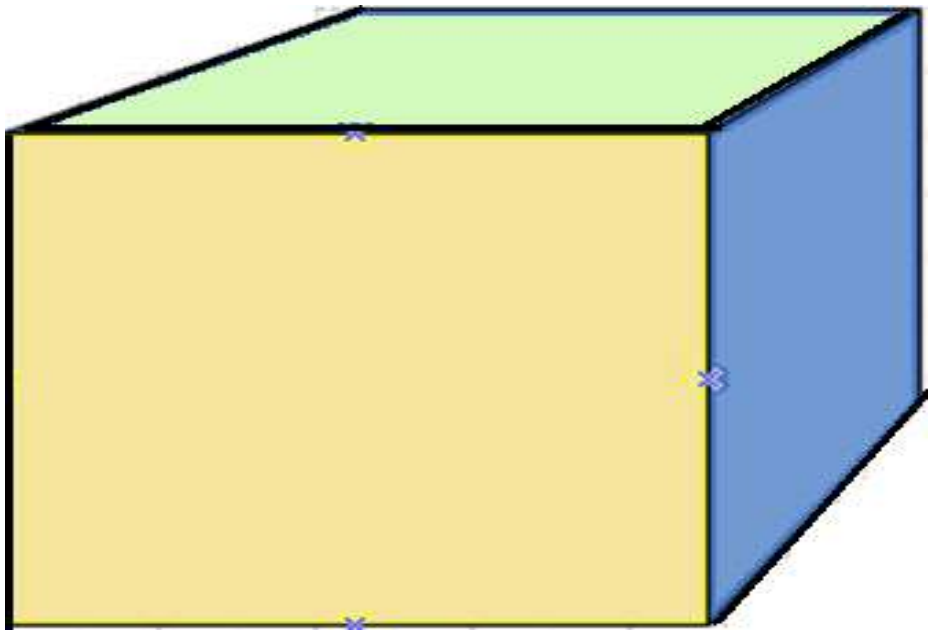


Figure 7.22 Dimensional modeling life cycle.

Understanding Dimension – Cube

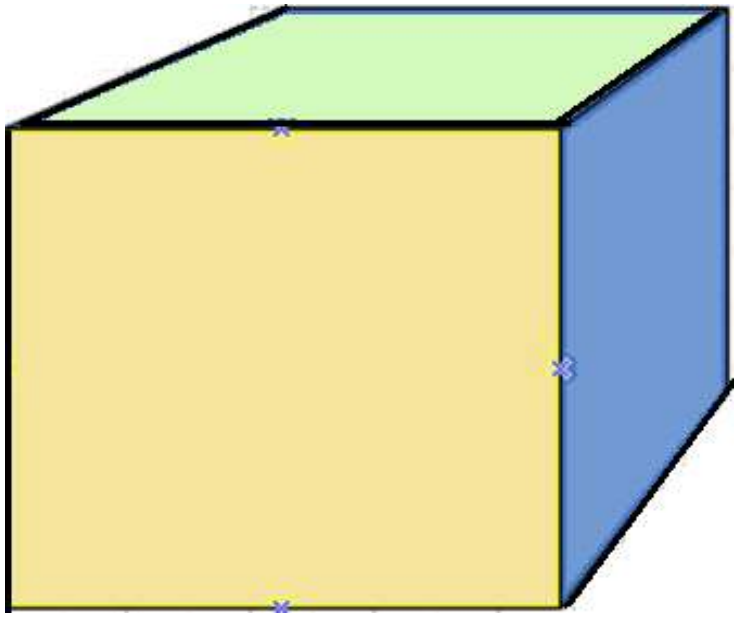
- An extension to the two-dimensional Table.
- For example in the previous scenario CEO wants a report on revenue generated by different services across regions during each quarter



Understanding Dimension – Cube (contd.)

Dimension Hierarchy

Grain
Fact



N .America

Europe

Asia Pacific

Testing

Consulting

Production Support

Q1 Q2 Q3 Q4

Recap (contd.)

- Difference between OLTP and OLAP

	OLTP	OLAP
Definition	On Line Transaction Processing	On Line Analytical Processing
Data	Dynamic (day to day transaction / operational data)	Static (historical data)
Data Atomicity	Data is stored at microscopic level	Data is aggregated or summarized and stored at the higher level
Normalization	Normalized Databases to facilitate insertion, deletion and updation	De-normalized Databases to facilitate queries and analysis
History	Old data is purged or archived	Historical data stored to enable trend analysis and future predictions
Queries	Simple queries and updates Queries use small amounts of data (one record or a few records) Example: update account balance enroll for a course	Complex queries Queries use large amounts of data Example: Total annual sales for north region Total monthly sales for north region

Comparison of OLTP and DSS

OLTP Capability Examples

- Search & locate student(s)
- Print student scores
- Filter students above 90% marks
- Update student Grade
- Group by Batch and compute average score
- Find top 10 high performance students

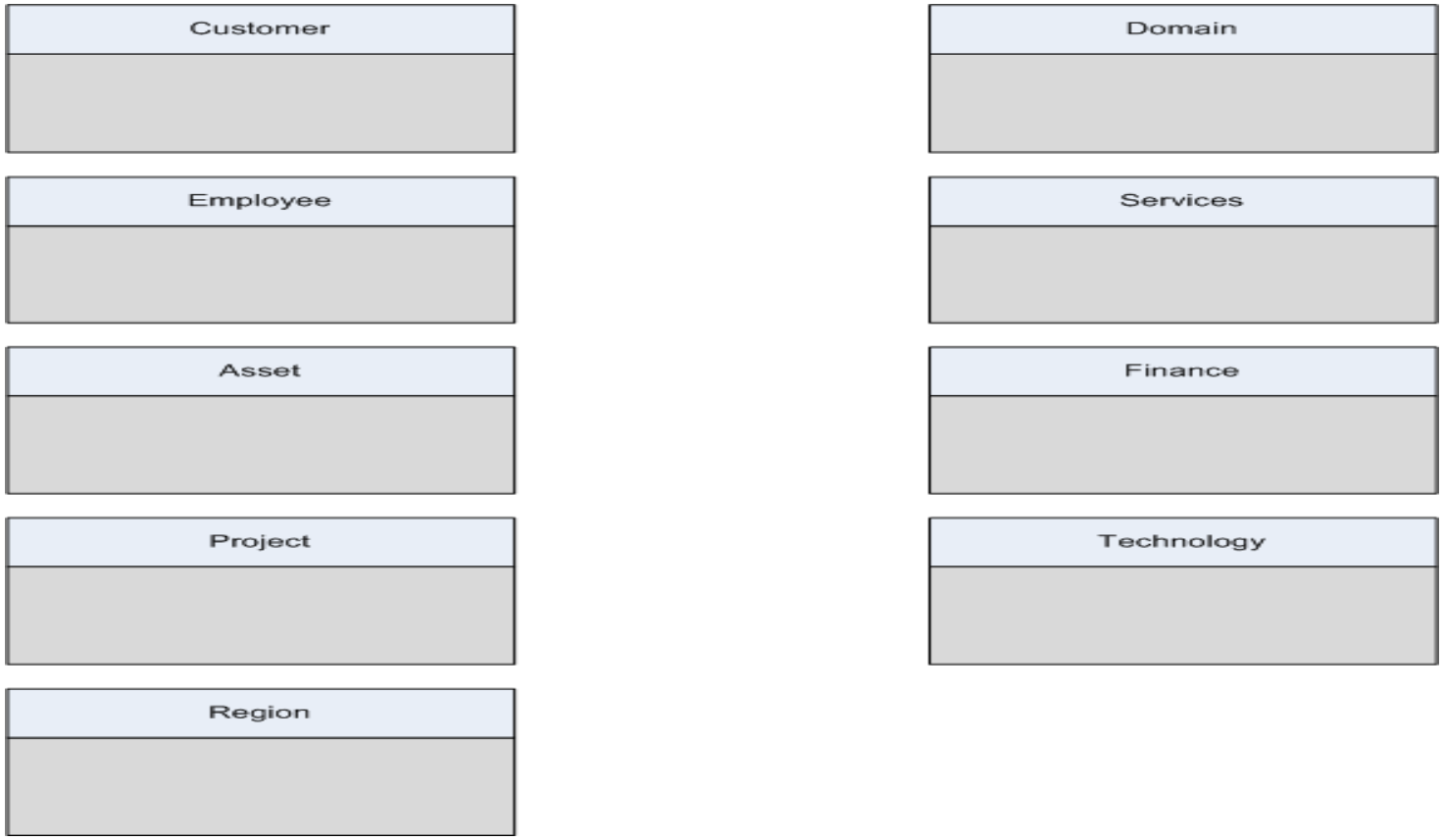
DSS Capability Examples

- Which courses have productivity impact on-the-job?
- Which colleges need to be rewarded for supplying students with consistent high on-the-job performance?
- What is the customer satisfaction improvement due to extended training?
- How project level profitability is influenced by certification?
- How much training is needed on future technologies for non-linear growth in BI?

Still a little fuzzy about what OLAP can do?

Introduction to On-line Analytical Processing (OLAP)

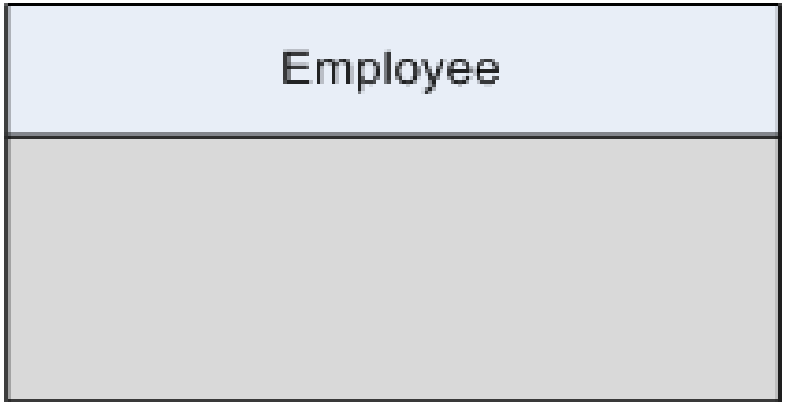
Scenario:
Internal systems department at Infosys maintains all relevant data in a database.
Conceptual schema is as shown below.



OLAP Contd.

- CEO of the company wants the following information from the IS department.
    - Number of employees added in the role of the company during the last quarter/6 months/1 year

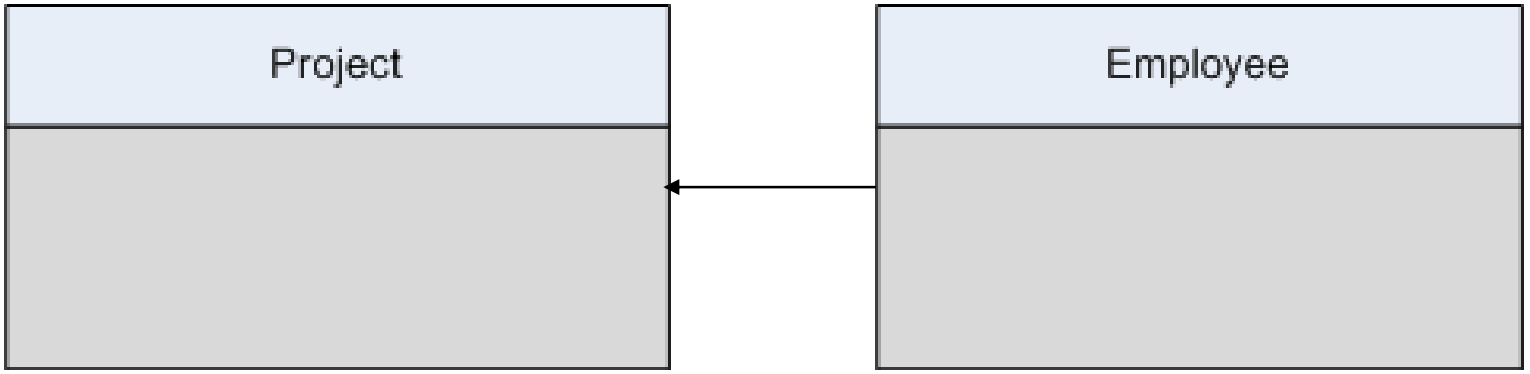
Q1. How many table(s) is/are required ?



OLAP Contd.

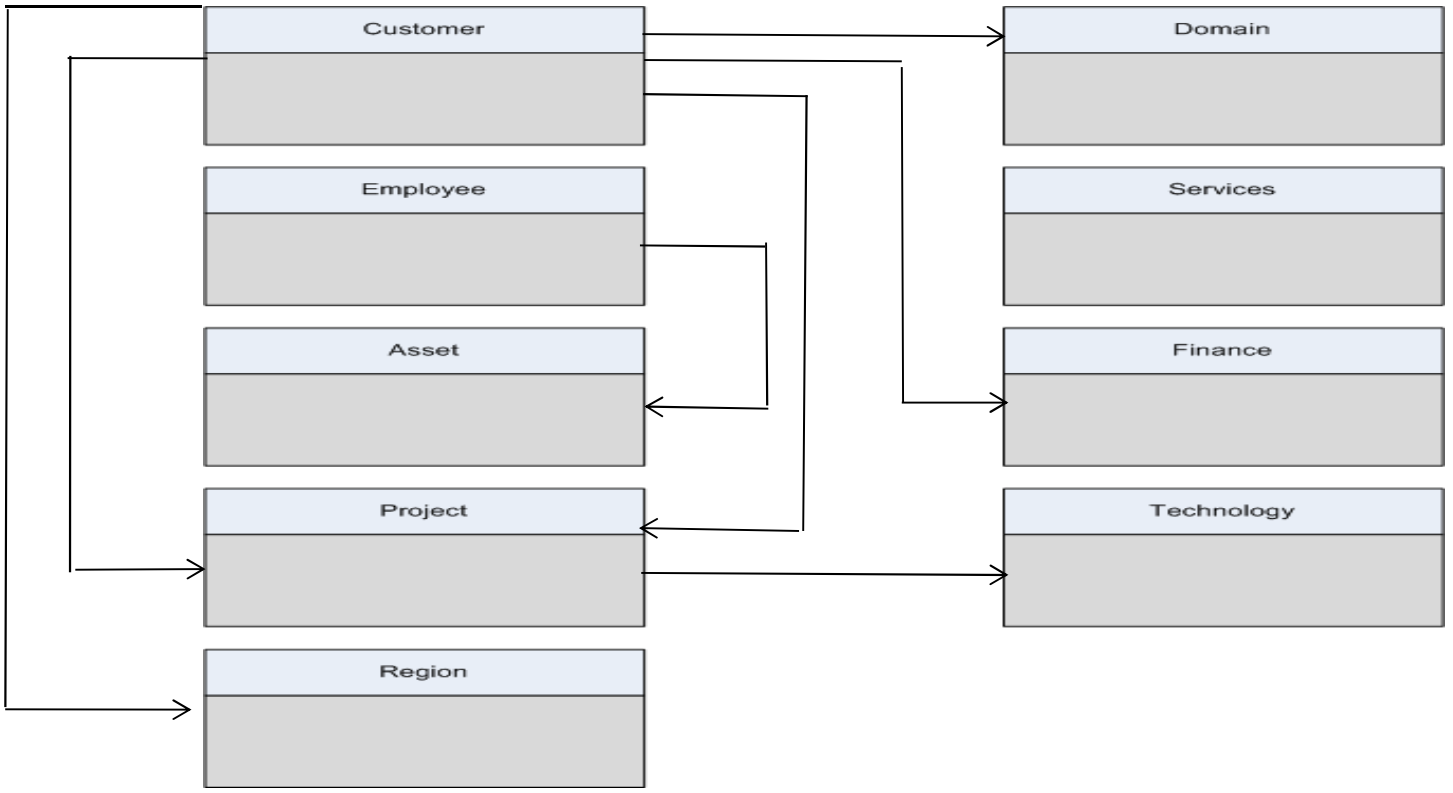
Q2. How many employees are currently in the projects ?

Q3. How many are on bench?



OLAP Contd.

Q4. Which **customer** from a **region** has given maximum **business** during the previous quarter on a **domain** under specific **technology** and who are the **PMs** of the **project** **and** **assets** **owned by them.**



SOLUTION?

Answer a Quick Question

So if there were very few updates and more of data-retrieval queries being made on your database, **what do you think would be a better schema to adopt?**

OLTP or OLAP

Introduction to Dimensional Modeling (DM)

- DM is a logical design technique used in Data Warehouses (DW). It is quite directly related to OLAP systems
- DM is a design technique for databases intended to support end-user queries in a DW
- It is oriented around understandability, as opposed to database administration

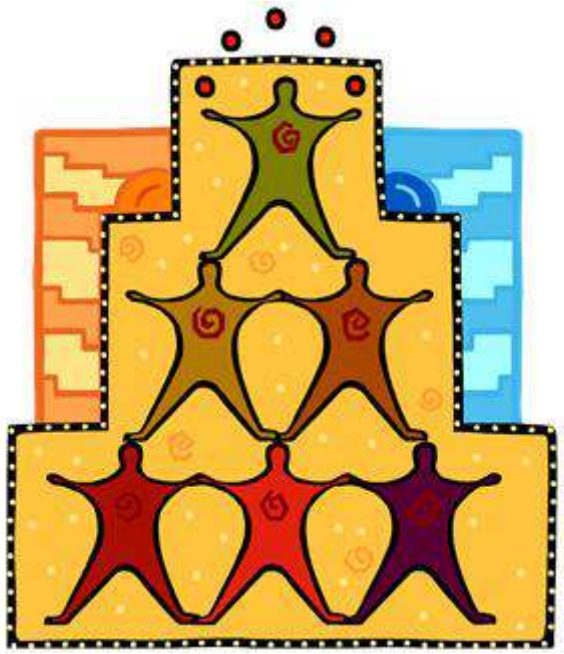
However, before we actually jump into MDDM…

let’s first understand the language of
Dimensional Modeling

MDDM Terminology

- Grain
- Fact
- Dimension
- Cube
- Star
- Snowflake

Of hierarchies and levels…



What Is a Grain?

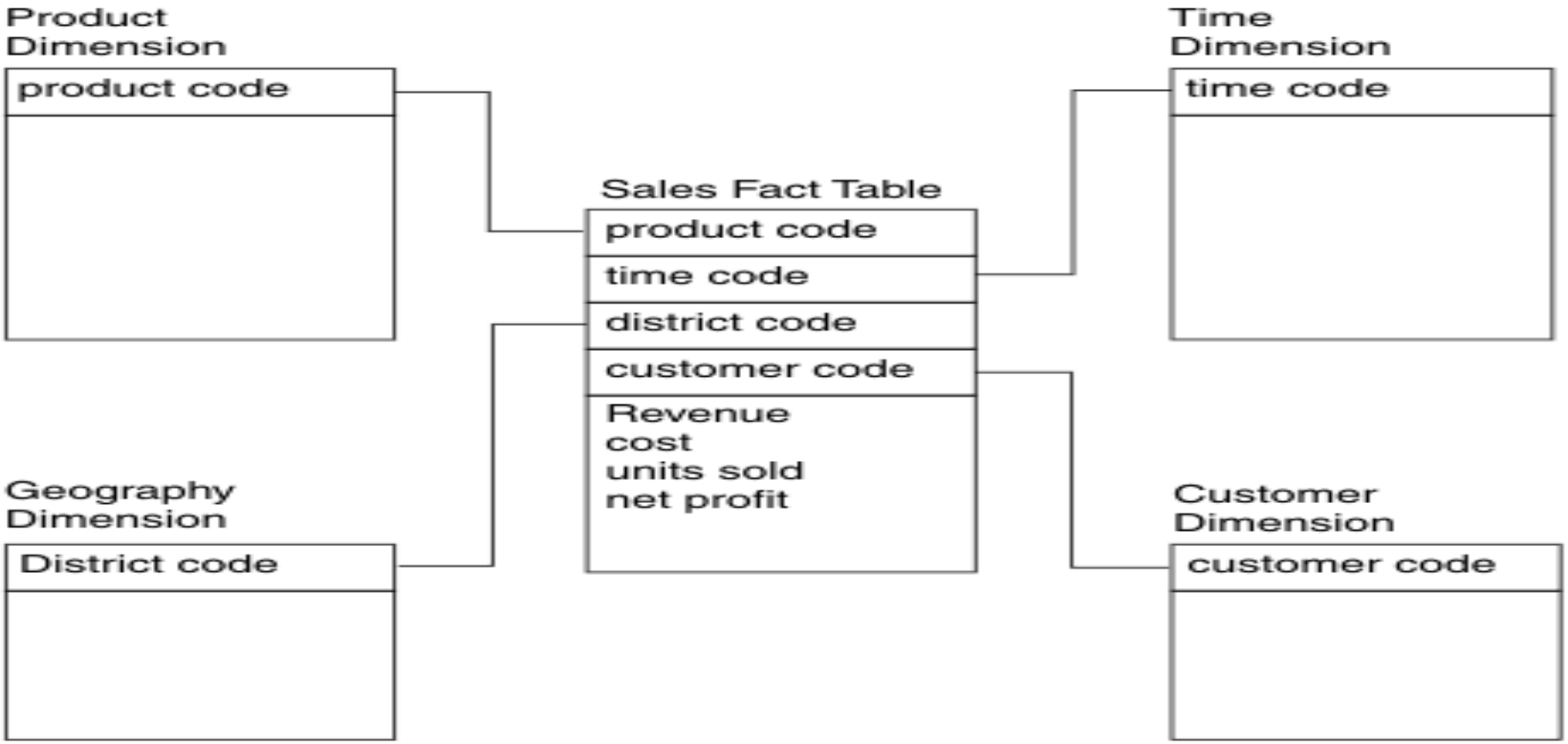
- Identifying the grain also means deciding the level of detail that will be made available in the dimensional model
- *Granularity* is defined as the detailed level of information stored in a table
- The more the detail, the lower is the level of granularity
- The lesser the detail, higher is the level of granularity

What can be measured, can be controlled…

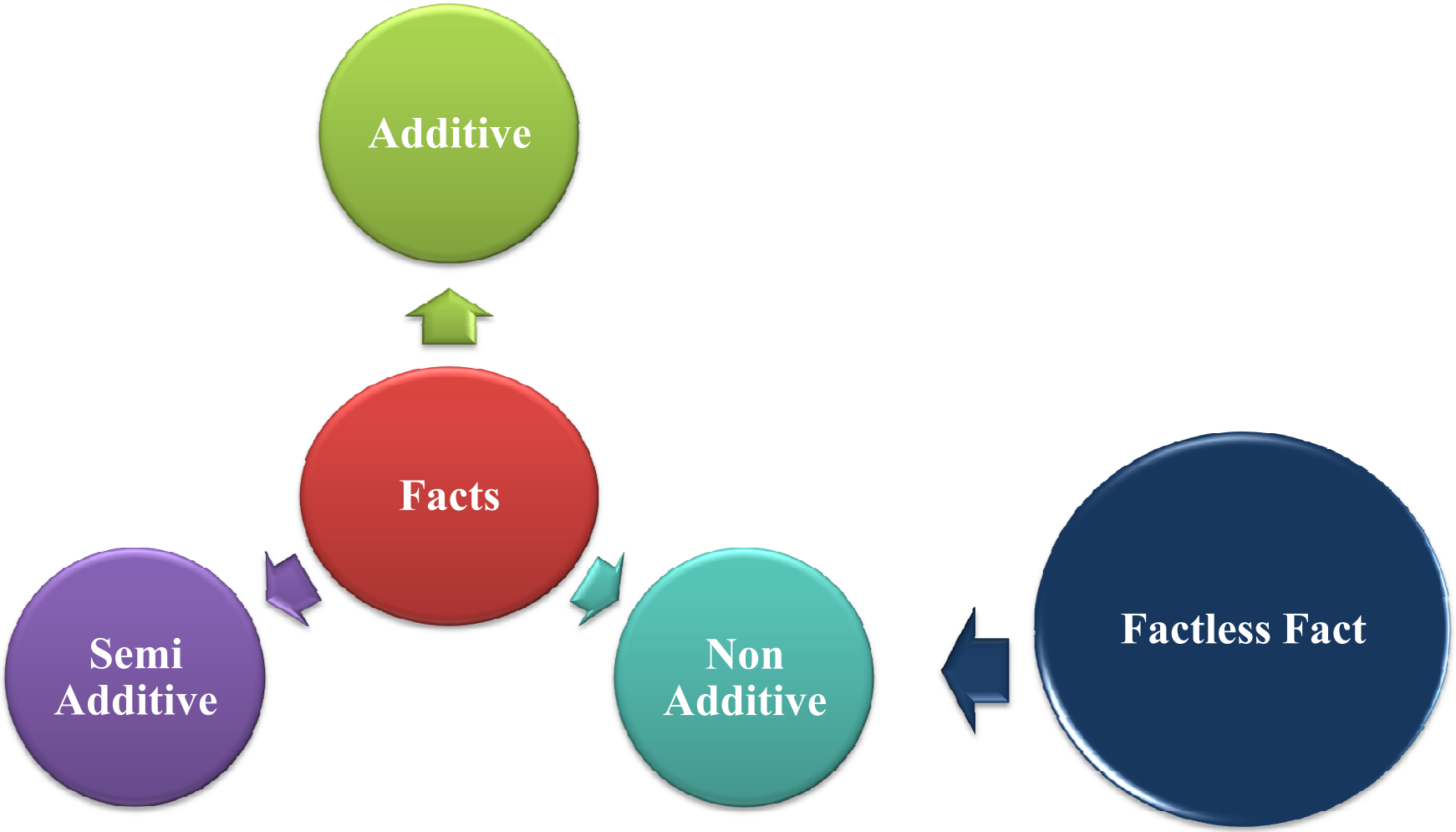
…and do you know how such measurements are stored in a data warehouse?

Facts and Fact Tables

- Consists of at least two or more foreign keys
- Generally has huge numbers of records
- Useful facts tend to be numeric and additive



Fact Types



And what about descriptive data?

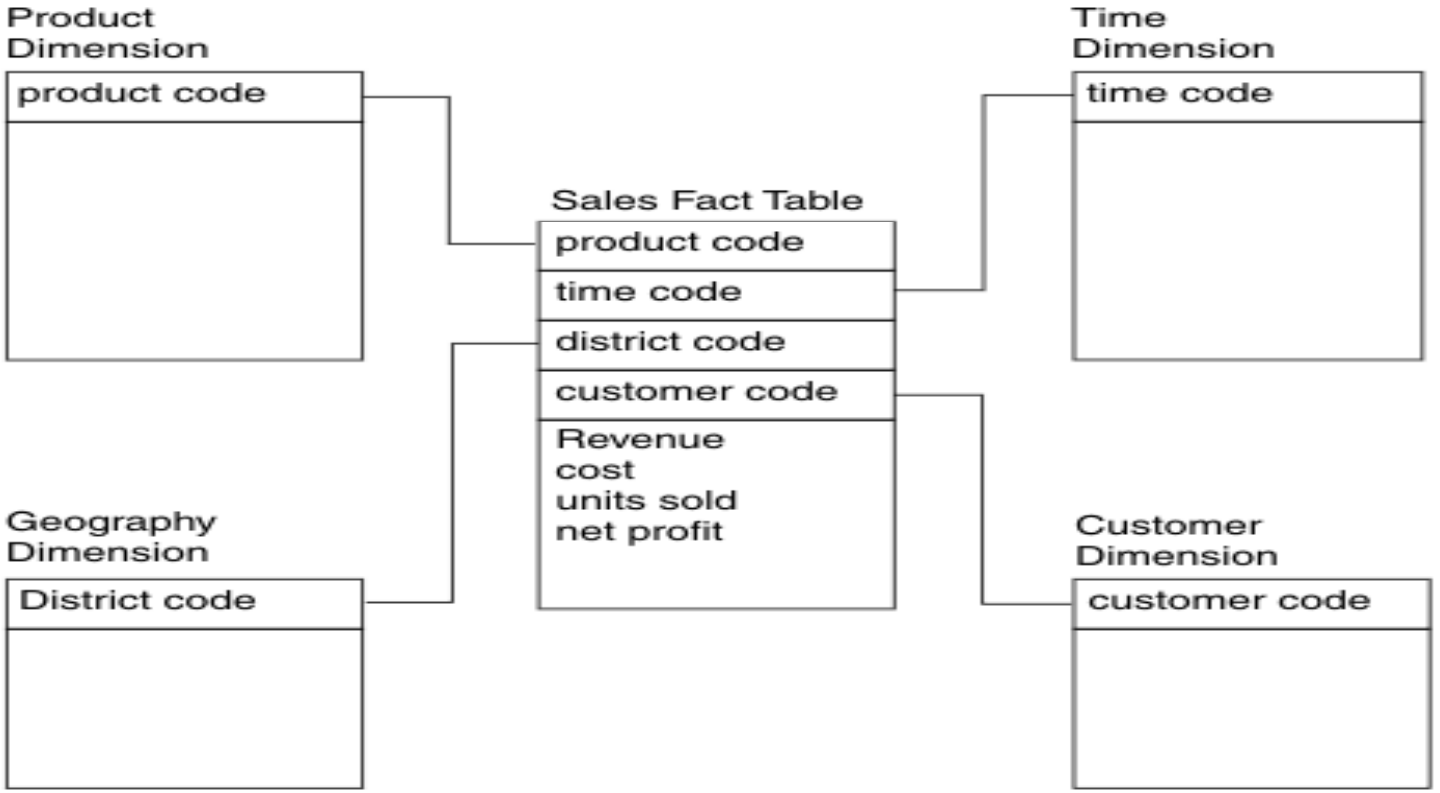
Answer a Quick Question

Can a data warehouse schema take different forms with respect to normalization?

Star Schema

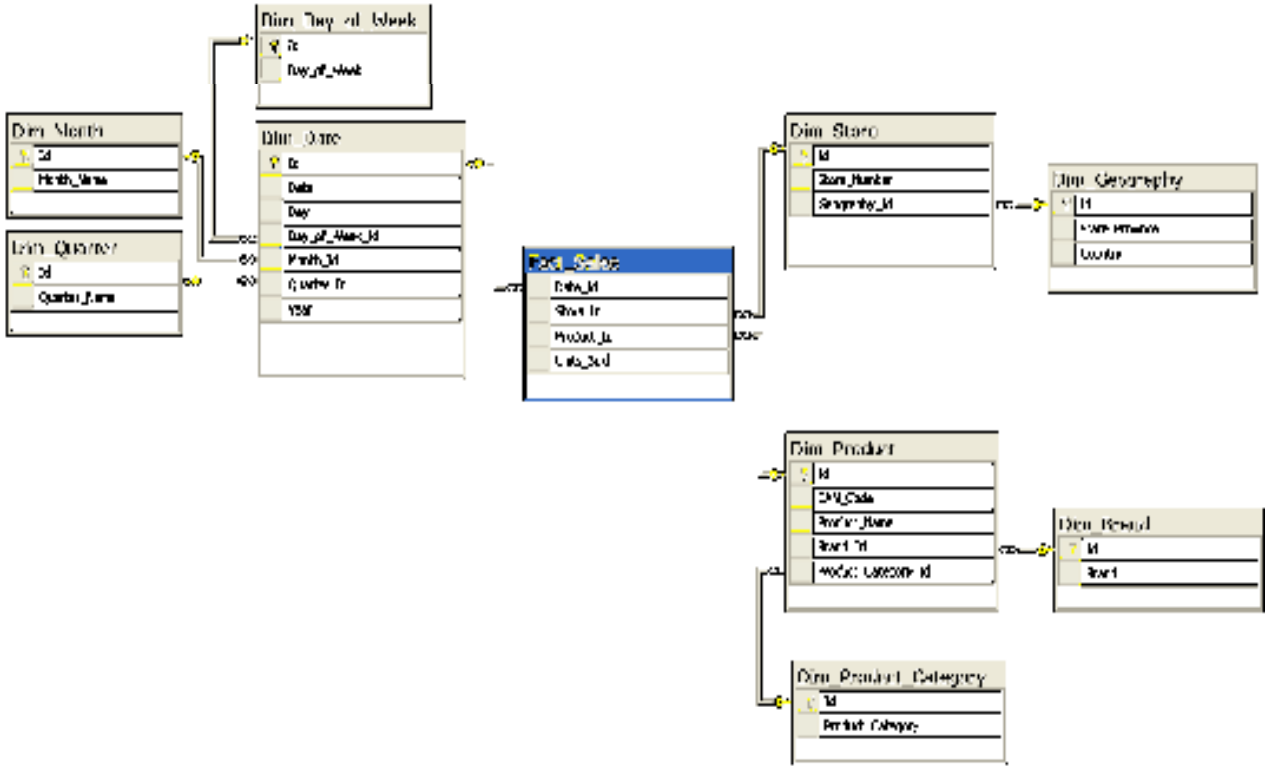
The basic star schema contains four components.
These are:

Fact table, Dimension tables, Attributes and Dimension hierarchies



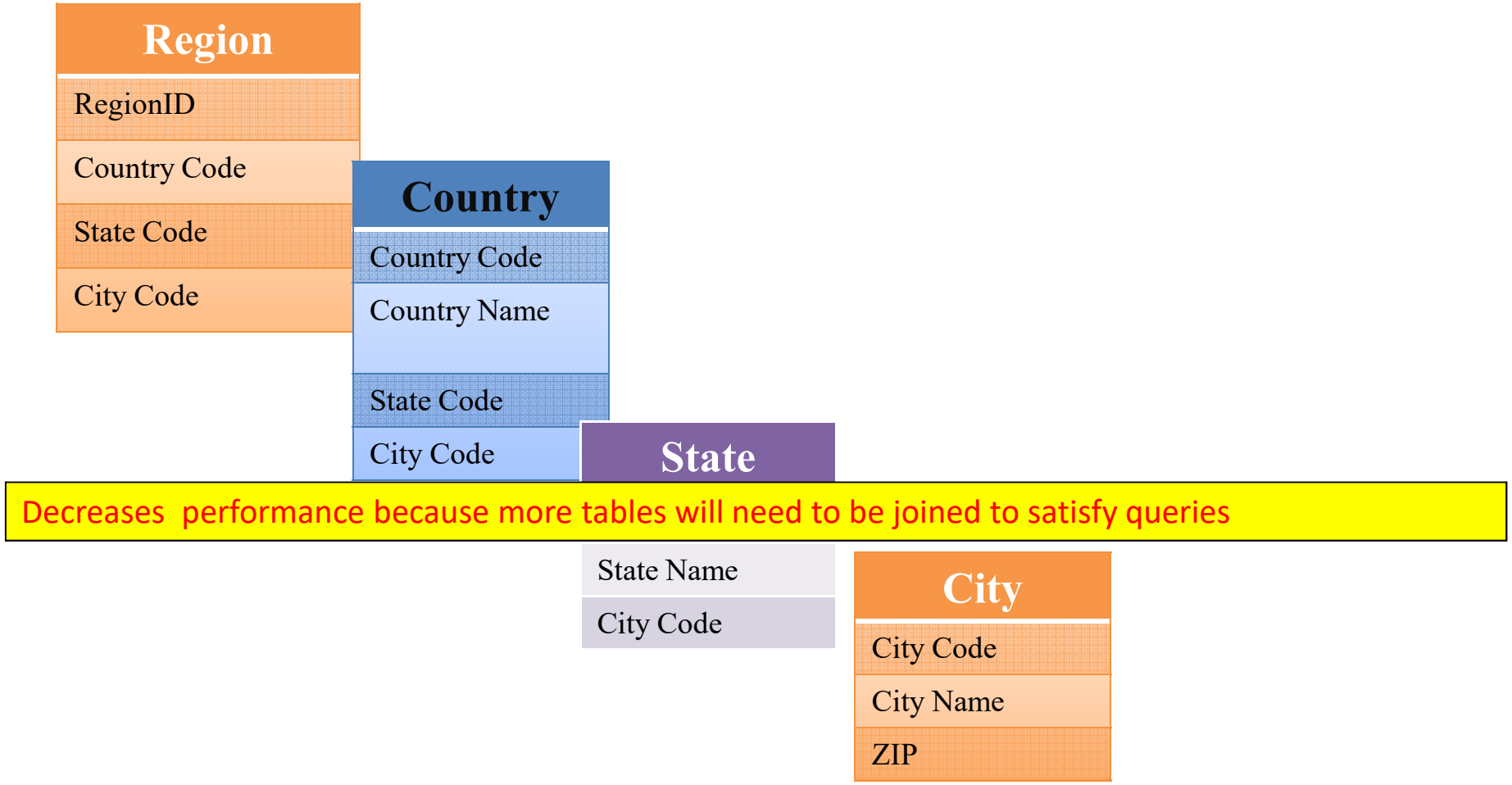
Snow Flake Schema

- Normalization and expansion of the dimension tables in a star schema result in the implementation of a snowflake design.
- A dimension table is said to be snow flaked when the low-cardinality attributes in the dimension have been removed to separate normalized tables and these normalized tables are then joined back into the original dimension table.



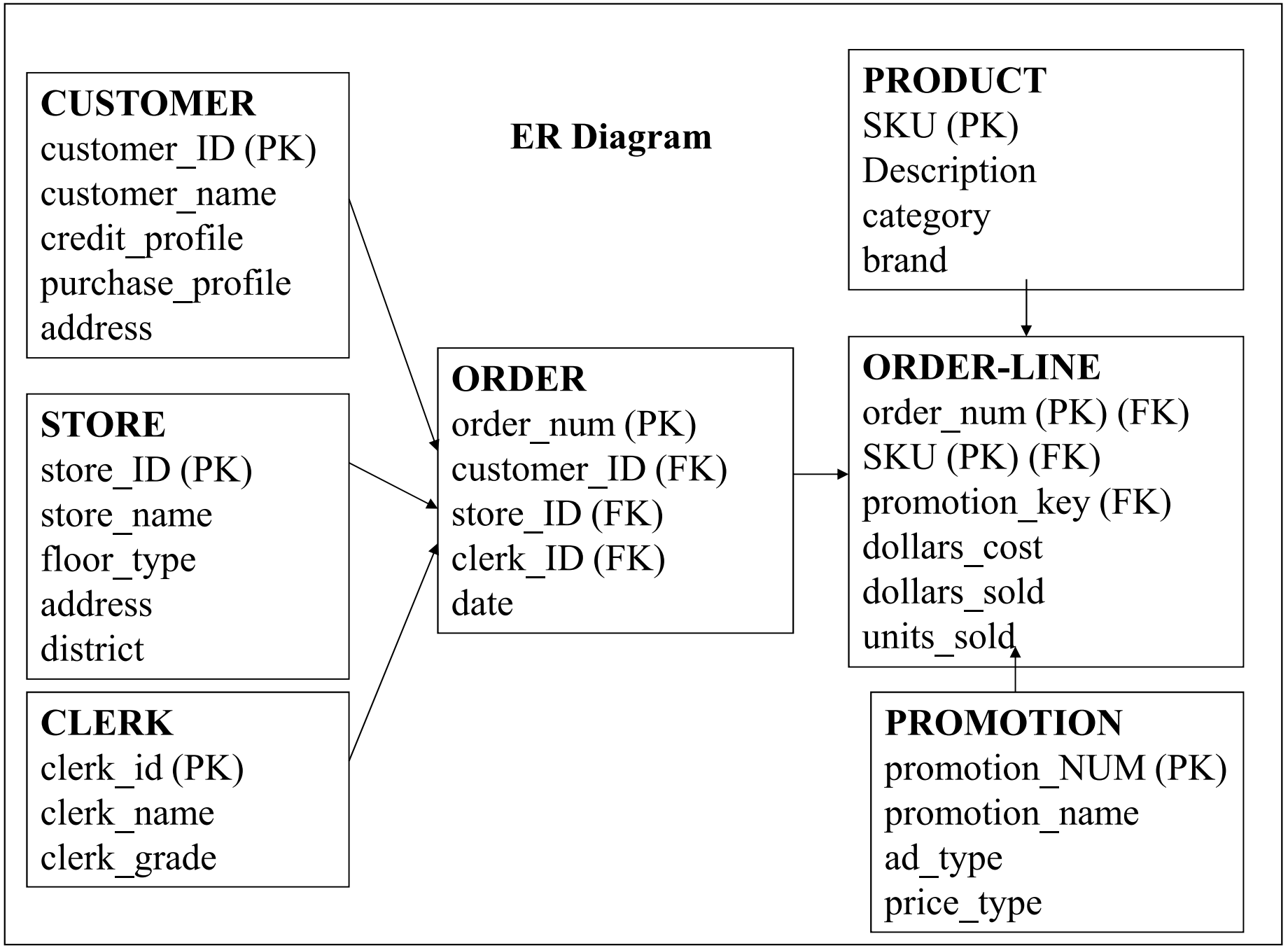
Snow Flaking Example

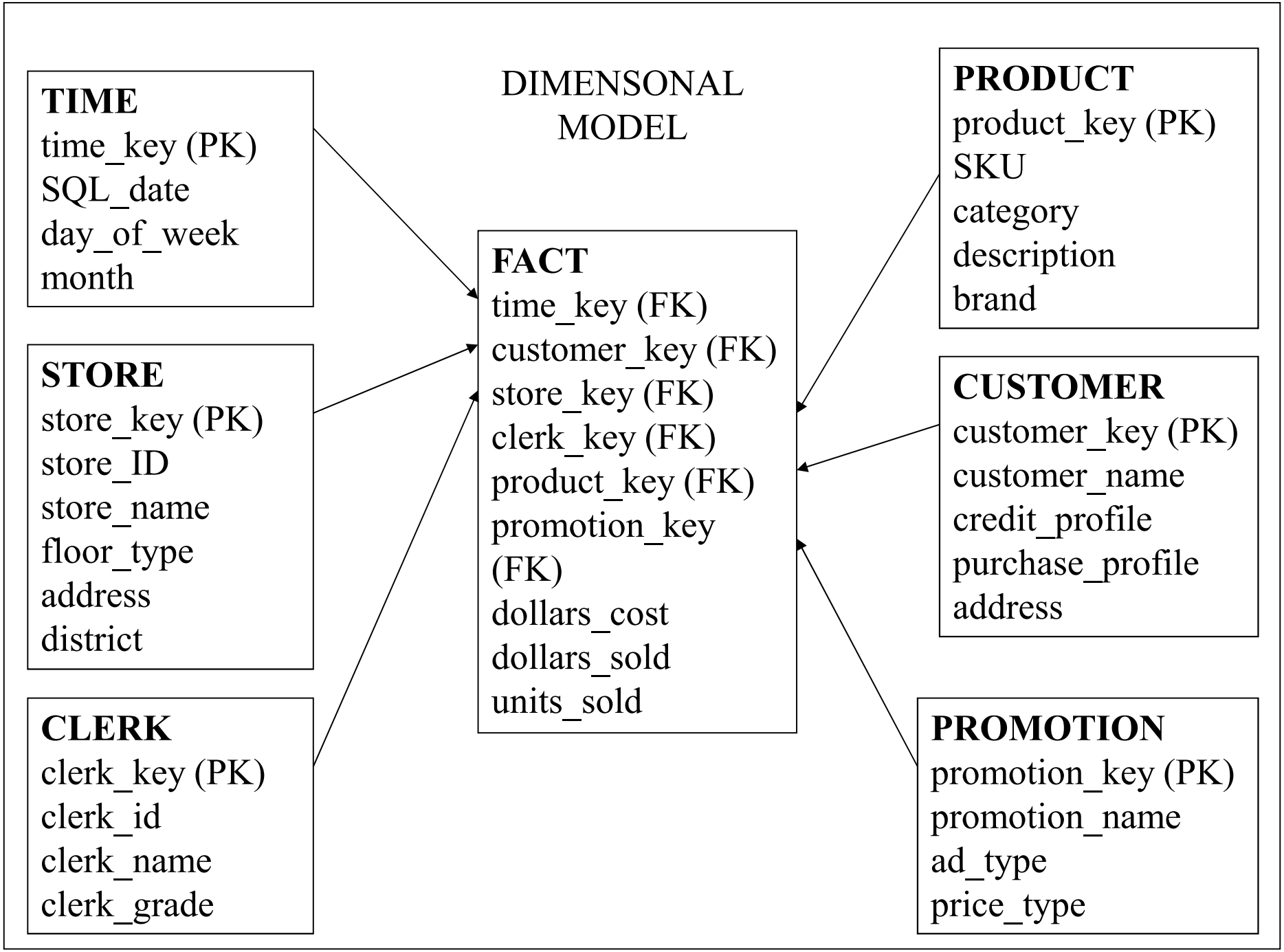
- Consider the Normalized form of Region dimension



Armed with these weapons that we call ‘Concepts’, let’s step into the battlefield!

Case Study
Conversion of a ER Model to a Dimensional Model


ER Diagram
CUSTOMER
customer_ID (PK)
customer_name
credit_profile
purchase_profile
address
STORE
store_ID (PK)
store_name
floor_type
address
district
CLERK
clerk_id (PK)
clerk_name
clerk_grade
ORDER
order_num (PK)
customer_ID (FK)
store_ID (FK)
clerk_ID (FK)
date
PRODUCT
SKU (PK)
Description
category
brand
ORDER-LINE
order_num (PK) (FK)
SKU (PK) (FK)
promotion_key (FK)
dollars_cost
dollars_sold
units_sold
PROMOTION
promotion_NUM (PK)
promotion_name
ad_type
price_type


DIMENSONAL MODEL
TIME
time_key (PK)
SQL_date
day_of_week
month
STORE
store_key (PK)
store_ID
store_name
floor_type
address
district
CLERK
clerk_key (PK)
clerk_id
clerk_name
clerk_grade
FACT
time_key (FK)
customer_key (FK)
store_key (FK)
clerk_key (FK)
product_key (FK)
promotion_key (FK)
dollars_cost
dollars_sold
units_sold
PRODUCT
product_key (PK)
SKU
category
description
brand
CUSTOMER
customer_key (PK)
customer_name
credit_profile
purchase_profile
address
PROMOTION
promotion_key (PK)
promotion_name
ad_type
price_type

Summary

- Basics of Database
- OLTP
- MDDM
- Cube
- Star Schema
- Snowflake schema

Food for Thought!

1. Who according to you would be the user of OLAP?
2. Who would need the Multidimensional perspective of data?