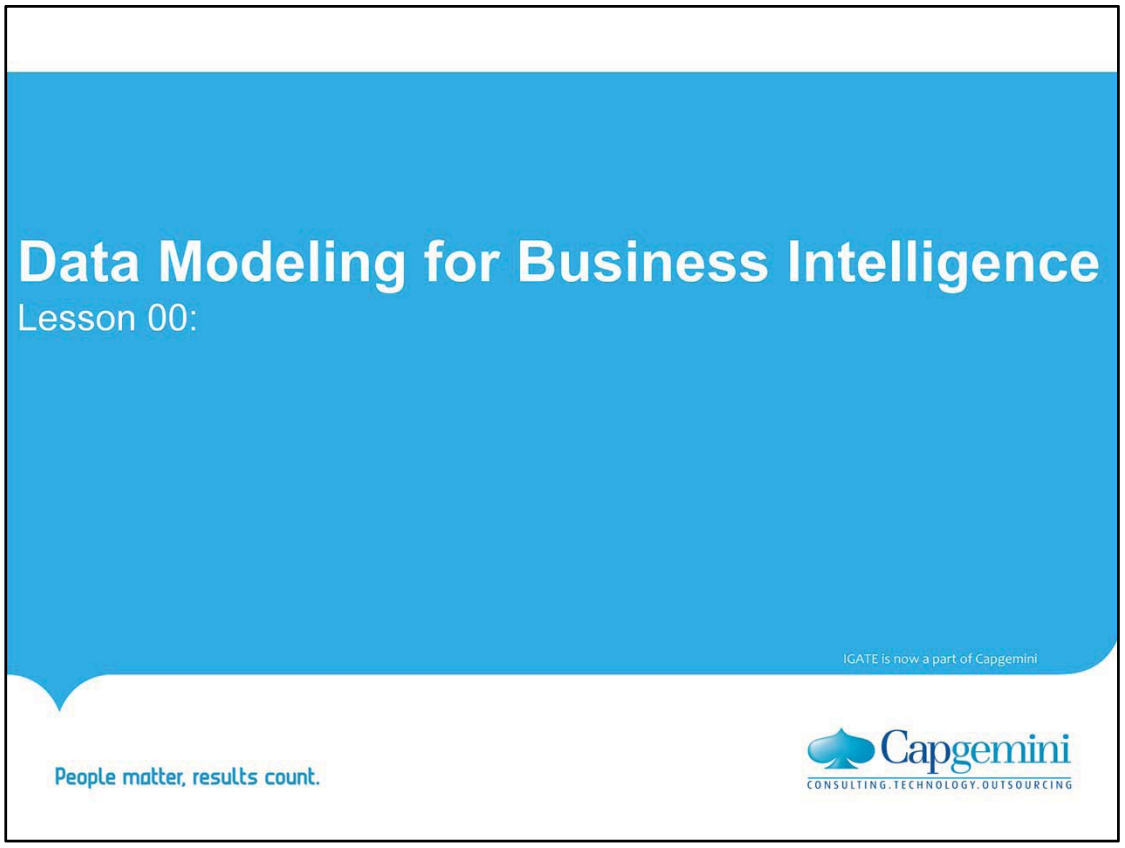


Document History

Date	Course Version No.	Software Version No.	Developer / SME	Change Record Remarks
June 2011	0.1D	NA	Vandana Mistry	Content Creation
June 2016	0.2D	NA	Swati Rao	Material Revamp as per Integrated ToC for I & D LoT

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Course Goals and Non Goals

- Course Goals
  - At the end of this program, participants gain an understanding of basic concepts in Data Modeling.
- Course Non Goals
  - Implementation of Data Modeling tools.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

Pre-requisites

- Fair knowledge of DW concepts

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Intended Audience

- Software Engineers and Senior Software Engineers

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Day Wise Schedule

- Day 1
  - Lesson 1: Introduction to Data Modeling
  - Lesson 2: Understanding Business Requirements
  - Lesson 3: Conceptual Model
  - Lesson 4: Logical Model

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

References

- Student material:
  - Class Book (presentation slides with notes)
- Book:
  - Data Modeling techniques for data warehousing
- Web-site:
  - http://www.datawarehouse.org

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Next Step Courses (if applicable)

- BI related tool training

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Other Parallel Technology Areas

- NA

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Data Modeling for Business Intelligence

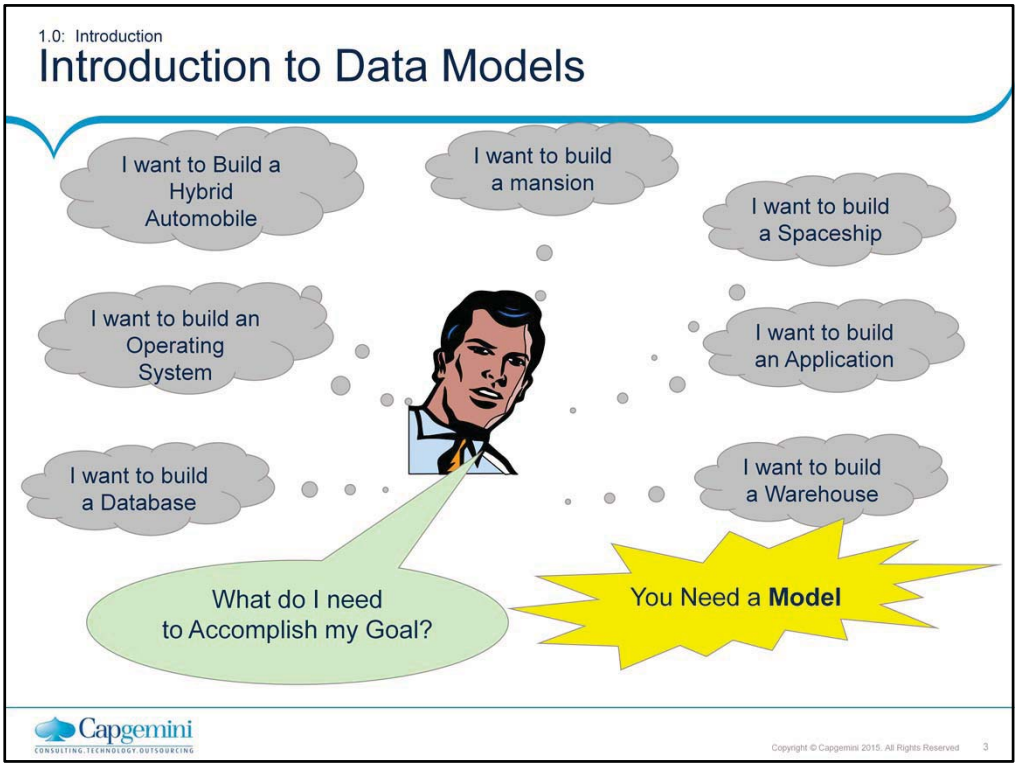
Lesson 1: Introduction to Data Modeling

Lesson Objectives

- On completion of this lesson on Data Modeling, you will be able to:
  - State the importance of data modeling
  - Identify features of a good data model
  - Identify who should be involved in data modeling
  - List the database design stages and deliverables
  - Explain classification of information

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

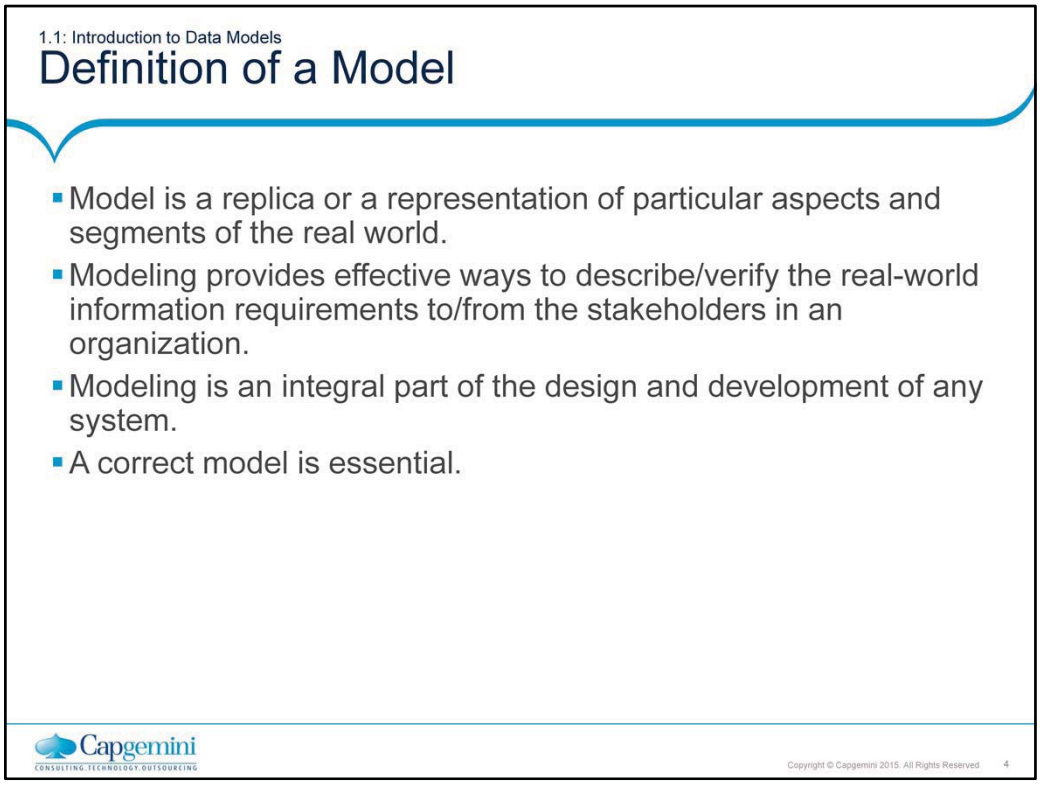
 2



Introduction:

In order to do any of the above, First, you need to create a model of the requirement.

Without a proper model of a requirements, an adequate system cannot be correctly designed and implemented. A good model of high quality forms an essential prerequisite for any successful system.



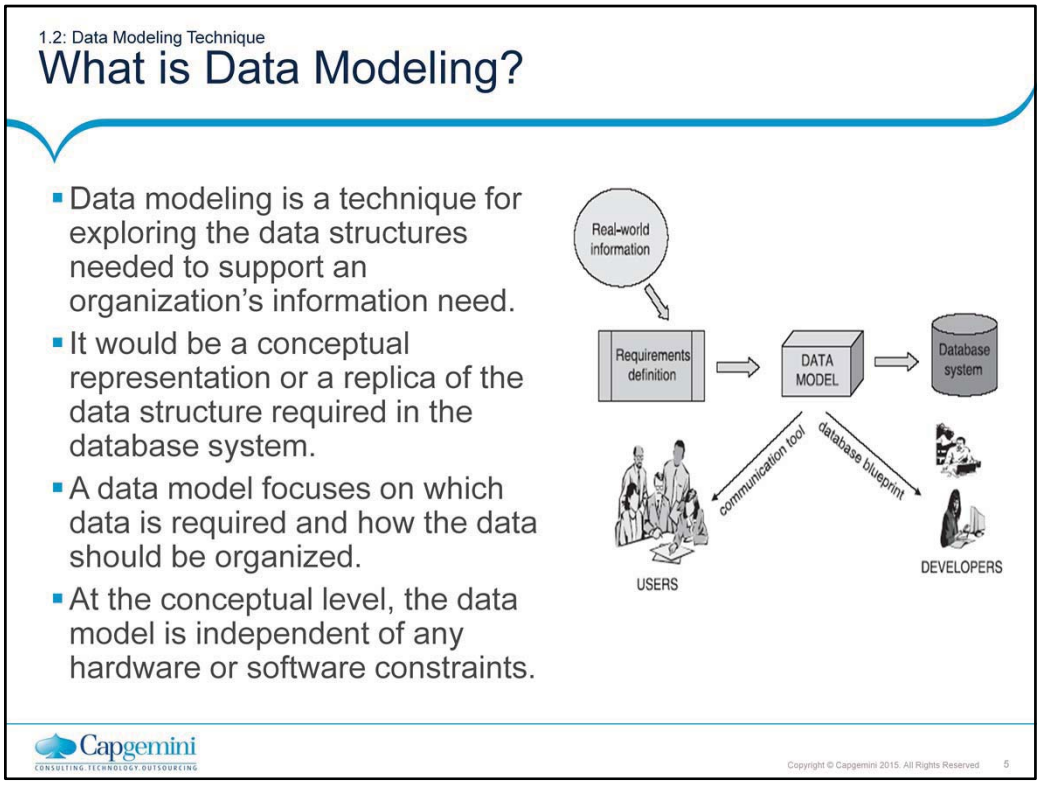
What is a model?

A model serves two primary purposes:

1) As a true representation of some aspects of the real world, a model enables clearer communication about those aspects.
2) A model serves as a blueprint to shape and construct the proposed structures in the real world.

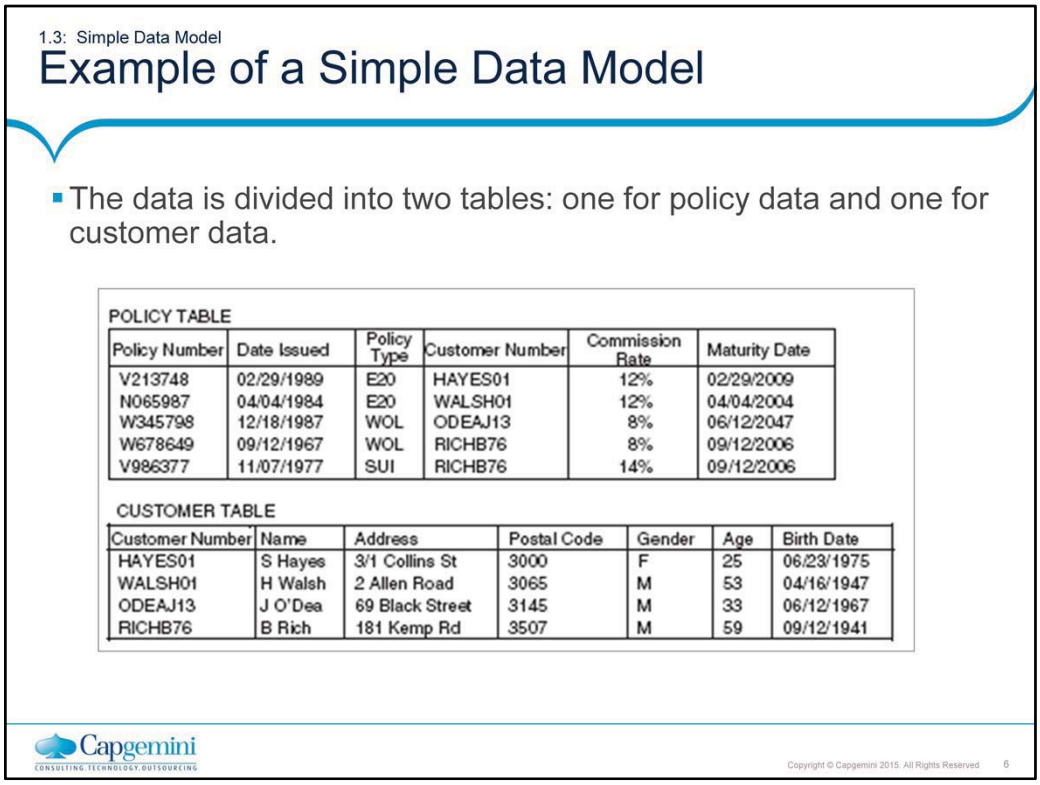
So, what is a data model? A data model is an instrument that is useful in the following ways:

1) A model helps the users or stakeholders clearly understand the database system that is being implemented. It helps them understand the system with reference to the information requirements of an organization.
2) It enables the database practitioners to implement the database system exactly conforming to the information requirements.



What is Data Modeling?

- At this level, the data model is generic; it does not vary whether you want to implement an object-relational database, a relational database, a hierarchical database, or a network database.
- At the next level down, a data model is a logical model relating to the particular type of database relational, hierarchical, network, and so on. This is because in each of these types, data structures are perceived differently.
- If you proceed further down, a data model is a physical model relating to the particular database management system (DBMS) you may use to implement the database.



A closer look at the model might suggest some questions:

- The meaning of customer is not clear whether he/she is the person insured or the beneficiary of the policy, or the person who pays the premiums?

Could a customer be more than one person, for example, a couple? If so, how would we interpret Age, Gender, and Birth Date?

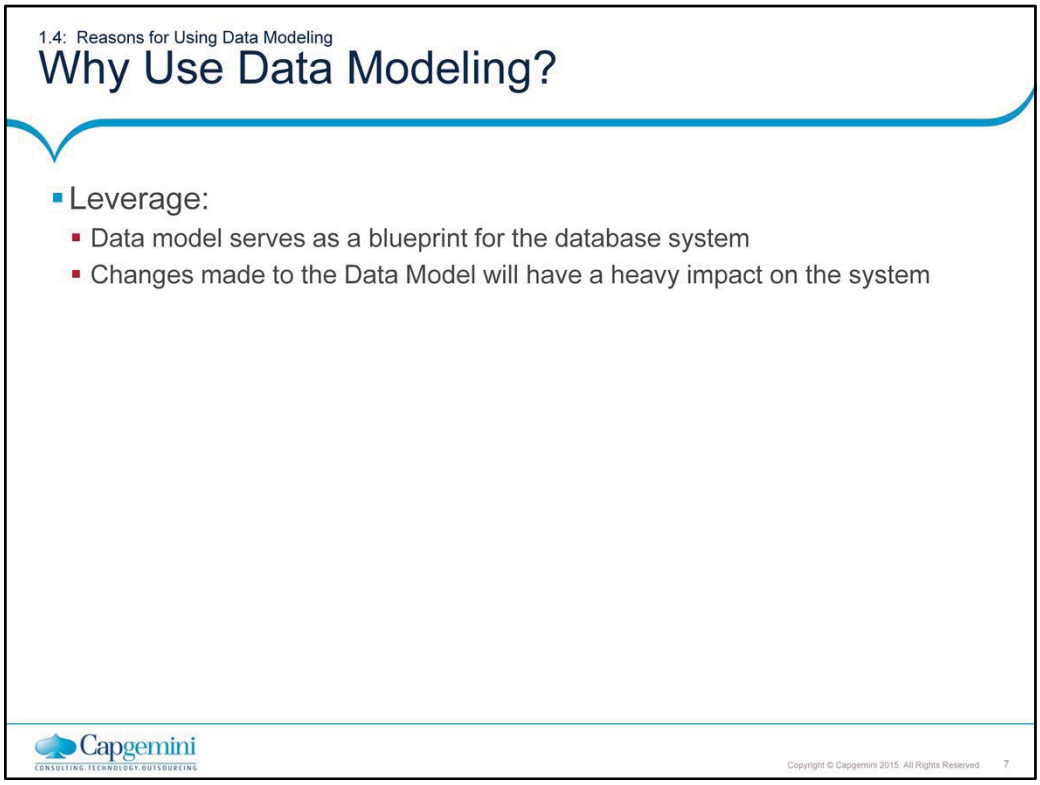
- There may not be any requirement for storing the customers ages. It will be easier to calculate it from Birthdate.
- Is there a relationship between Commission Rate and a Policy Type?. Do policies of type E20 always earn 12% commission?

This will imply recording the same rate many times. How do we record the Commission Rate for a new type of policy if we have not yet sold any policies of that type?

- Customer Number appears to consist of an abbreviated surname, initial,and a two-digit “tie-breaker” to distinguish customers who would otherwise have the same numbers.

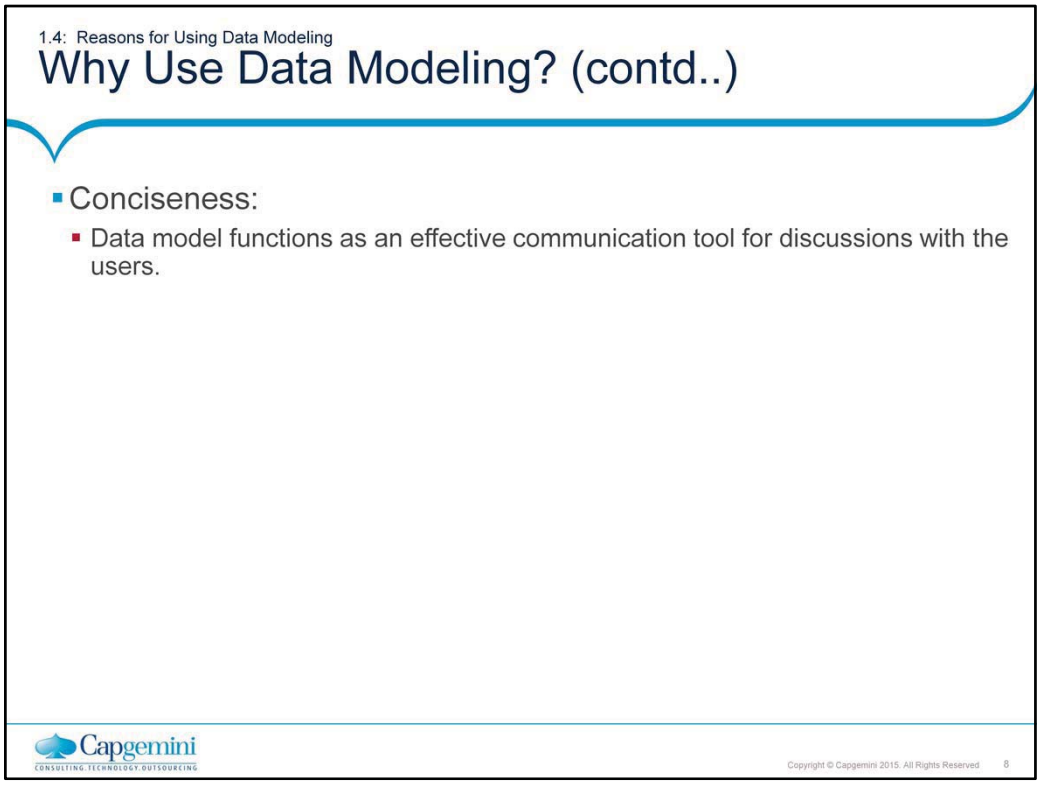
Is this a good choice?

- Would it be better to hold customers’ initials in a separate column from their family names?
- “Road” and “Street” have not been abbreviated consistently in the Address column. Should we impose a standard?



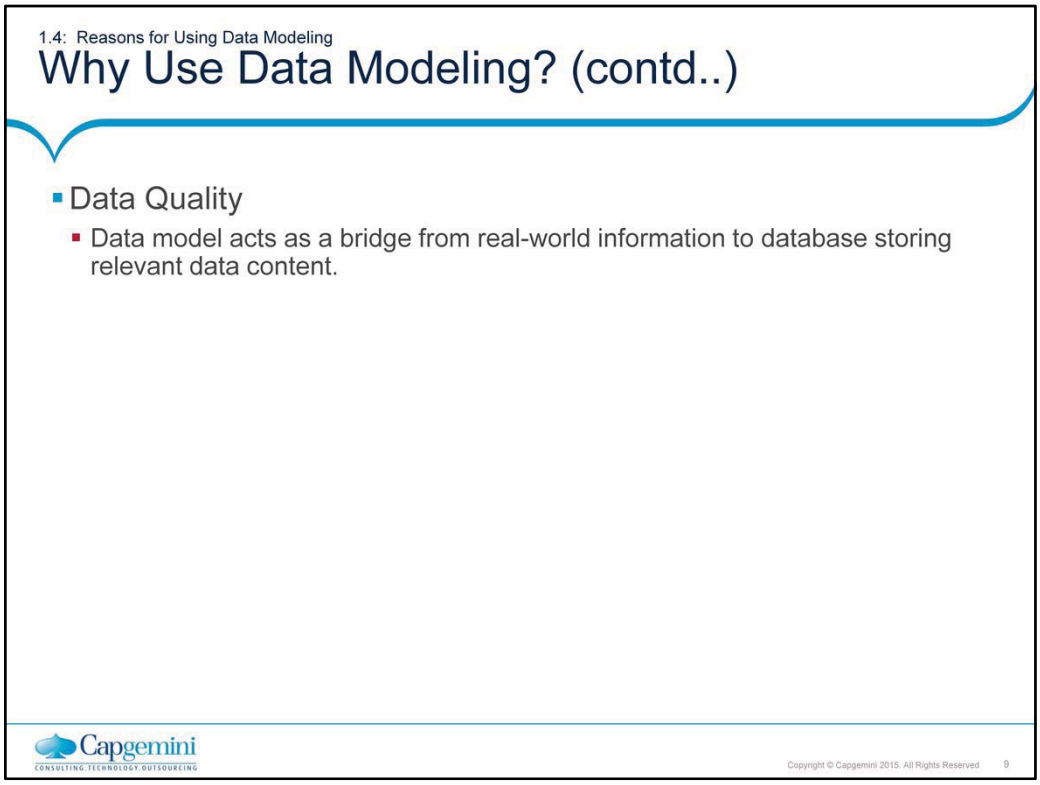
Why Use Data Modeling?

Leverage: The key reason for giving special attention to data organization is the leverage. A small change to a data model may have a major impact on the whole system. Therefore, you can opt for modifying the data model instead of the system. For the most commercial information systems, the programs are far more complex. Also, considerable time is consumed in specifying and constructing them, as compared to the database. However, their contents and structures are heavily influenced by the database design. In the insurance example, imagine that we need to change the rule that each customer can have only one address. The change to the data model may well be reasonably straightforward. Perhaps we will need to add a further two or three address columns to the Policy table. With modern database management software, the database can probably be reorganized to reflect the new model without much difficulty. But the real impact is on the rest of the system. Report formats will need to be redesigned to allow for the extra addresses; screens will need to allow input and display of more than one address per customer; programs will need loops to handle a variable number of addresses; and so on. Changing the shape of the database may in itself be straightforward, but the costs come from altering each program that uses the affected part. In contrast, fixing a single incorrect program, even to the point of a complete rewrite, is a (relatively) simple, contained exercise.



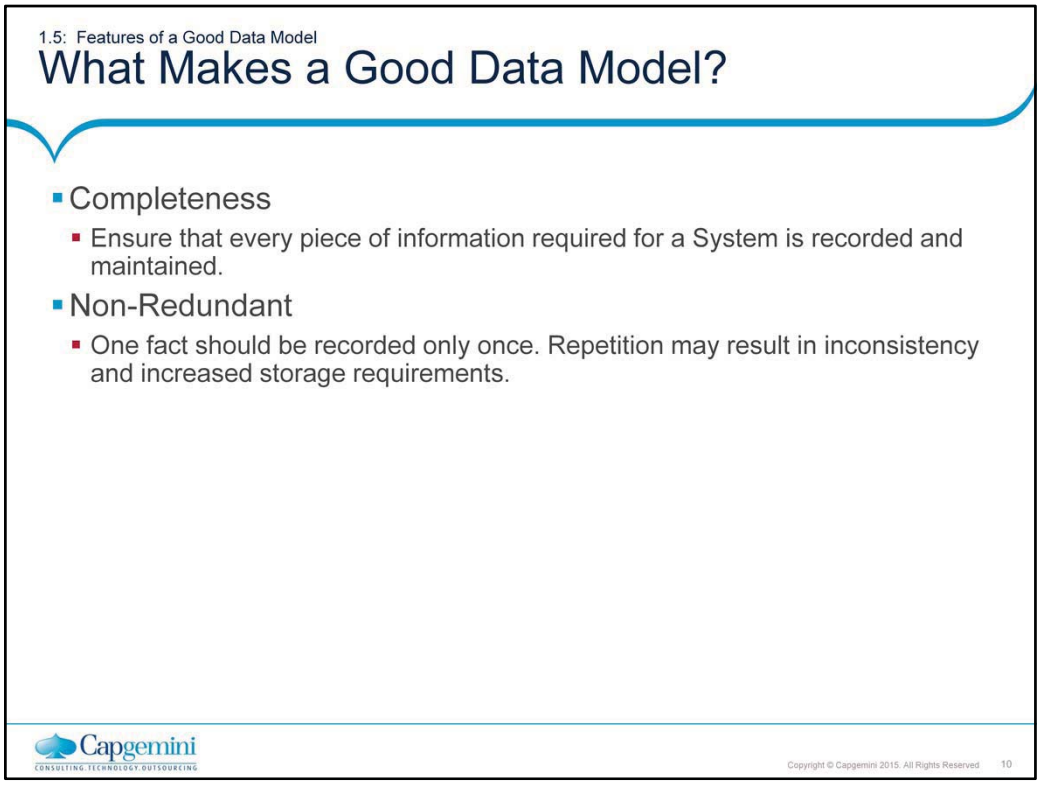
Why Use Data Modeling?

- **Conciseness**: A data model is a very powerful tool for establishing requirements and capabilities of information systems. Its valuable because of its *conciseness*. It implicitly defines a whole set of screens, reports, and processes needed to capture, update, retrieve, and delete the specified data. The data modeling process can tremendously facilitate our understanding of the essence of business requirements.



Why Use Data Modeling?

- **Data Quality**: The data held in a database is usually a valuable business asset built up over a long period. Inaccurate data (poor **data quality**) reduces the value of the asset and can be expensive or impossible to correct. Frequently, problems with data quality can be traced back to a lack of consistency in (a) defining and interpreting data, and (b) implementing mechanisms to enforce the definitions.
  In the insurance example, is Birth Date in U.S. or European date format (mm/dd/yyyy or dd/mm/yyyy)? Inconsistent assumptions here by people involved in data capture and retrieval could render a large proportion of the data unreliable.

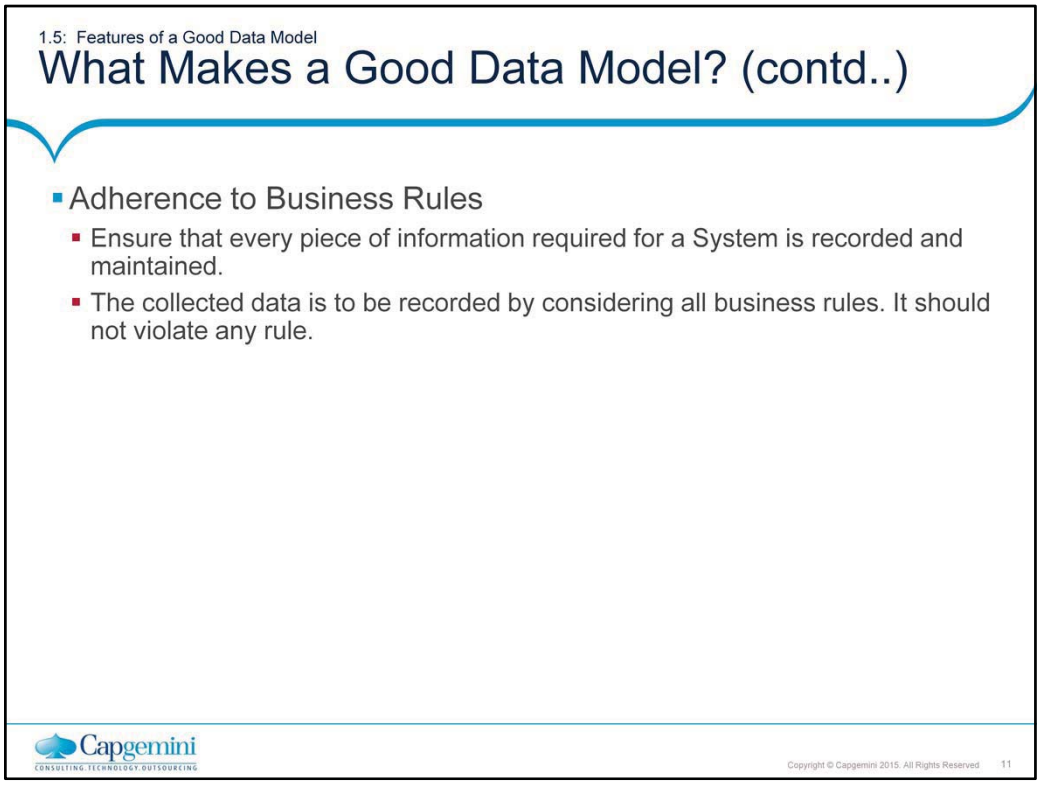


What Makes a Good Data Model?

•**Completeness**: The data model must support all the necessary data. A loss of small piece of information could result in significant loss to the company.

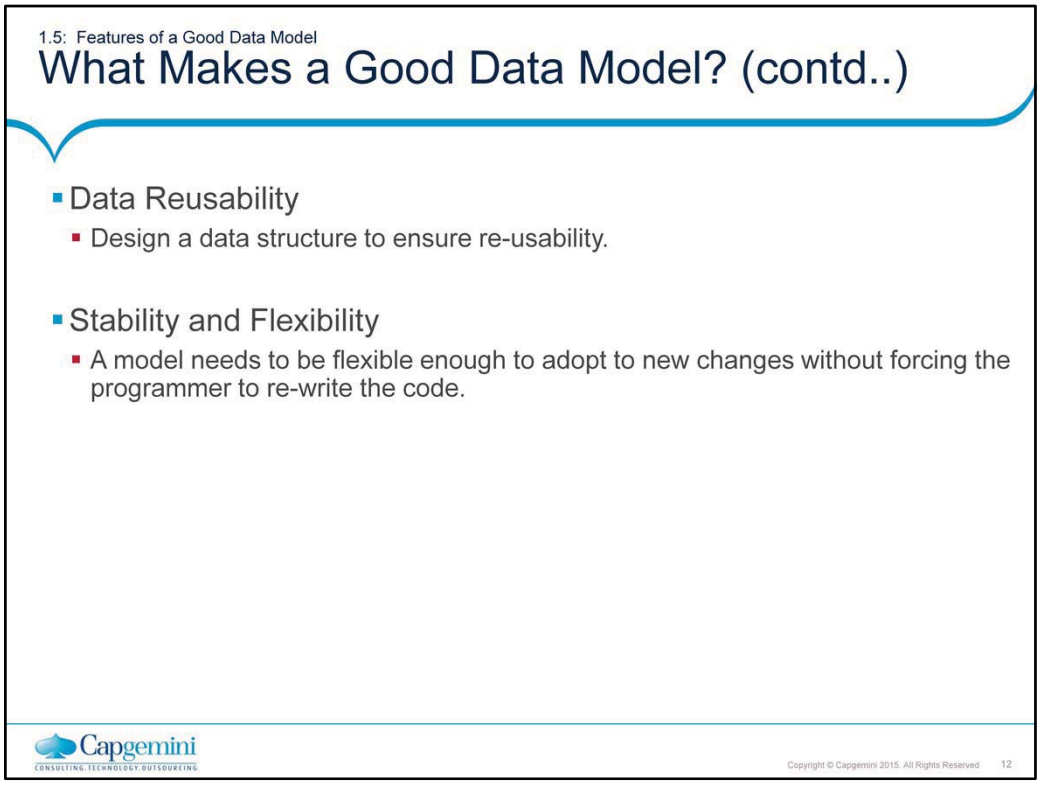
E.g. The insurance model lacks, does not have a column to record a customer's occupation and a table to record premium payments. If such data is required by the system, then these are serious omissions. Also, we have noted that we might be unable to register a commission rate if no policies had been sold at that rate.

•**Non-redundant**: Is the same information recorded more than once? In the example, the same commission rate could be held in many rows of the Policy table. The Age column records the same fact as Birth Date, in a different form. If we added another table to record insurance agents, we could end up holding data about people who happened to be both customers and agents in two places. Recording the same data more than once increases the amount of space needed to store the database.



What Makes a Good Data Model?

Adherence to Business Rules: The data model should accurately reflect and enforce the rules that apply to the business' data. The insurance model enforces the rule that each policy can be owned by only one customer, as there is provision for only one Customer Number in each row of the Policy table. No user or even programmer of the system will be able to break this rule: there is simply no place to record more than one customer against a policy (except extreme measures as holding a separate row of data in the Policy table for each customer associated with a policy). If this rule correctly reflects the business requirement, the resulting database will be a powerful tool in enforcing correct practice, and in maintaining data quality. On the other hand, any misrepresentation of business rules in the model may be very difficult to correct later (or to code around).

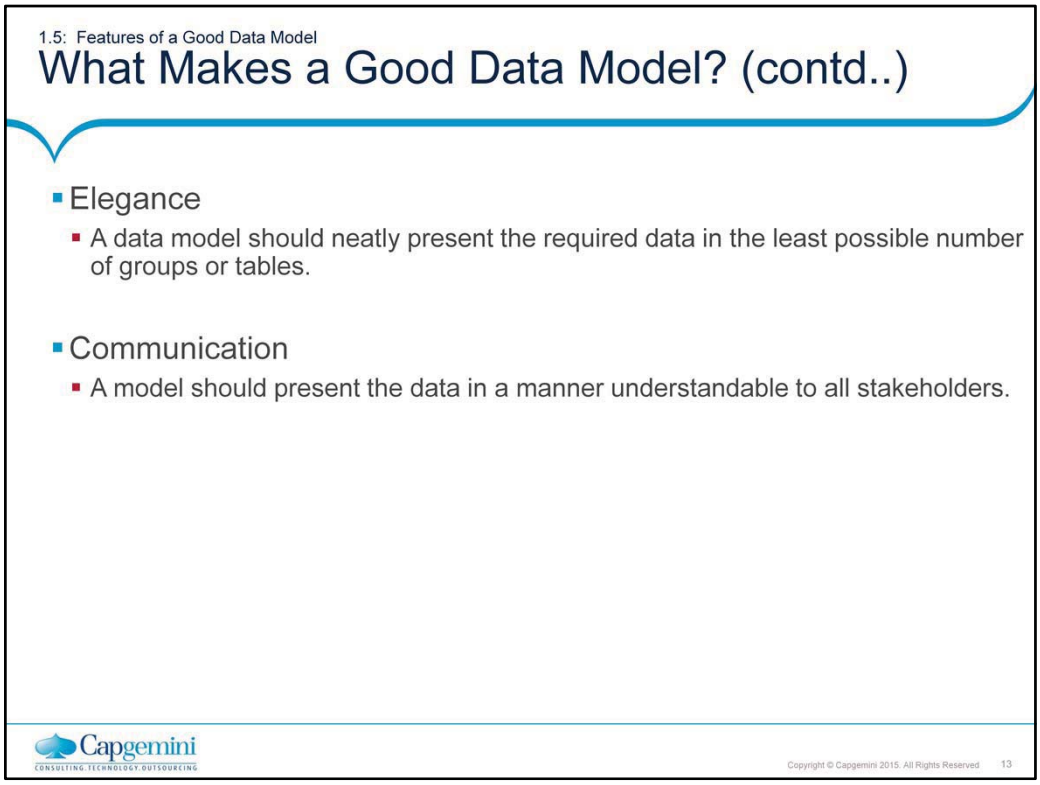


What Makes a Good Data Model? (contd.):

Data Reusability: The data stored in the database be should be reusable for purposes beyond those anticipated in the process model. Once an organization has captured data to for specific requirement, other potential uses and users emerge. An insurance company might initially record data about policies to support the billing function. The sales department then wants to use the data to calculate commissions; the marketing department wants demographic information; regulators require statistical summaries. Seldom can all of these needs be predicted in advance. If data has been organized with one particular application in mind, it is often difficult to use for other purposes. If the system users who have been into capture and storage of data are told that it cannot be made available to suit a new information requirement without extensive and costly reorganization, it could be very frustrating for them. Hence, as far as possible, data should be organized independently of any specific application.

•**Stability and Flexibility**: Regarding the stability and flexibility of the model, the following aspects should be considered:

- ➢Is the model able to cope with possible changes to the business requirements?
- ➢Are the existing tables able to accommodate any new data required to support such changes.
- ➢Alternatively, will simple extensions suffice?
- ➢Or else, will we be forced to make major structural changes, with corresponding impact on the rest of the system?
- ➢A data model is **stable** if we do not need to modify it at all, even if there is a change in requirements. A data model is **flexible** if it can be readily extended to accommodate probable new requirements with only minimal impact on the existing structure.

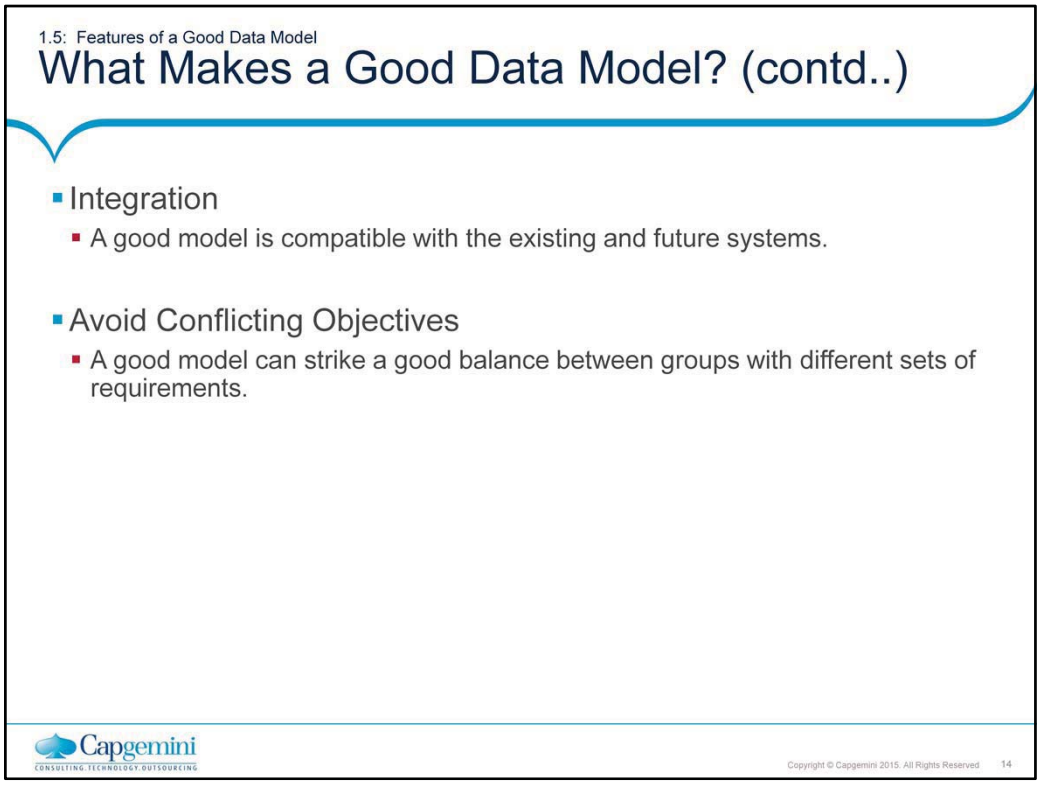


What Makes a Good Data Model? (contd.):

Elegance: Regarding the elegance of the model, the following aspect should be considered: Does the data model provide a reasonably neat and simple classification of the data? If the Customer table were to include only insured persons and not beneficiaries, we might need a separate Beneficiary table. To avoid recording facts about the same person in both tables, we would need to exclude beneficiaries who were already recorded as customers. Our Beneficiary table would then contain "beneficiaries who are not otherwise customers," an inelegant classification that would very likely lead to a clumsy system.

•**Communication**: Regarding the communication, the following aspects should be considered:

- ➢How effective is the model in supporting communication among the various stakeholders in the design of a system?
- ➢Do the tables and columns represent business concepts that the users and business specialists are familiar with and can easily verify?
- ➢Will programmers interpret the model correctly?



Features of a Good Data Model (contd.):

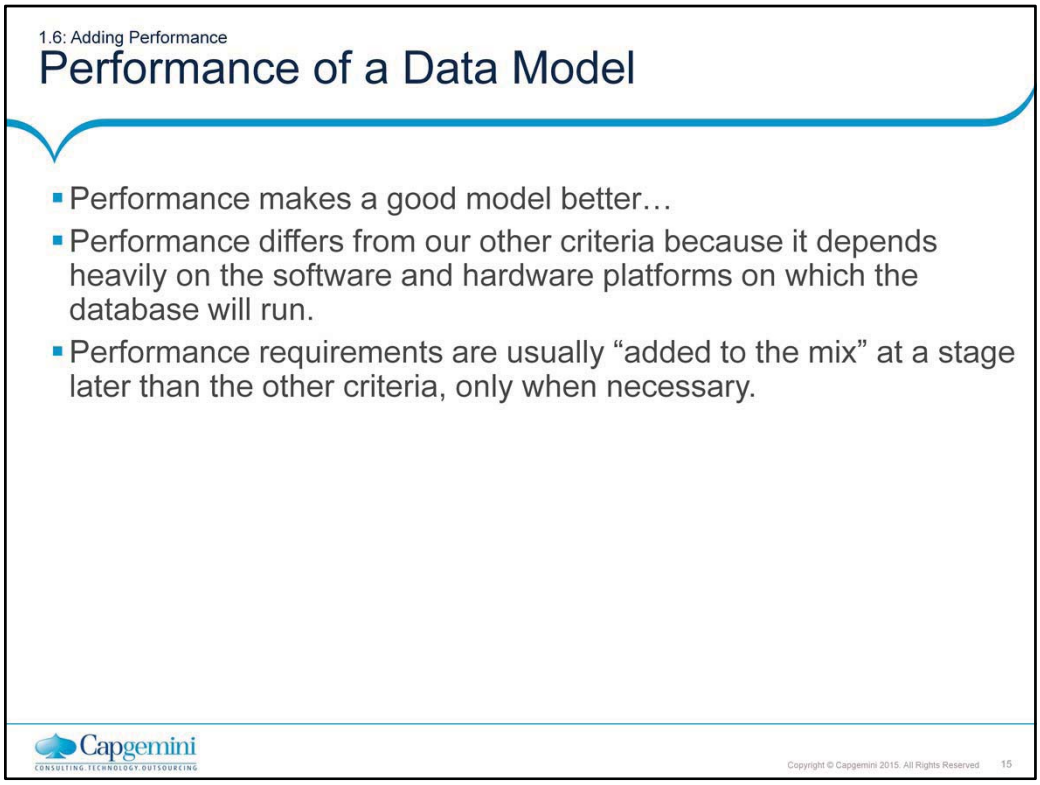
•**Integration**: Regarding integration of the model, the following aspect should be considered:

➢How will the proposed database fit in the organization's existing and future databases?

Even when individual databases are well designed, it is common for the same data to appear in more than one database and for problems to arise in collating together data from multiple databases.

•**Conflicting Objectives:** In many cases, the above aims conflict with one another. An elegant but radical solution may be difficult to communicate to conservative users. We may be so attracted to an elegant model that we exclude requirements that do not fit. A model that accurately enforces a large number of business rules will be unstable if some of those rules change. A model may be easy to understand because it reflects the perspectives of the immediate system users. However, it may not support reusability or may not integrate well with other databases.

Our overall goal is to develop a model that provides the best balance among these possibly conflicting objectives.



Considering the Performance at the initial stage could affect the natural process of data modeling. It may get biased towards a particular technology or database and make the design process biased.

Though it is a good idea to start considering about performance issues from the beginning, however, it must be just recorded for now and later implemented in physical design phase.

Performance can be introduced, once the logical model is freeze and selection of a particular technology & database is done. Since each technology will have it’s own methods to improve performance, the recorded requirement would be very handy while implementing the same.

Where Data Models are used ?

- Operational Systems
  - Traditional Applications designed to run the day-to-day business of the Enterprise
- External Systems ***
  - Data used within an Enterprise that is obtained from outside sources
- Staging Areas ***
  - Created to aid in the collection and transformation of data that is targeted for a Data Warehouse

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

Where Data Models are used ?

- Operational Data Store ***
  - W. H. Inmon and Claudia Imhoff definition: “A subject-oriented, integrated, volatile, current valued data store containing only corporate detailed data”.
- Data Warehouse (DW)
  - W. H. Inmon definition: “A subject-oriented, integrated, non-volatile, time-variant collection of data organized to support management needs”.
- Data Mart (DM)
  - TDWI definition: “A data structure that is optimized for access. It is designed to facilitate end-user analysis of data. It typically supports a single analytic application used by a distinct set of workers.”
- *** - Not discussed here

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

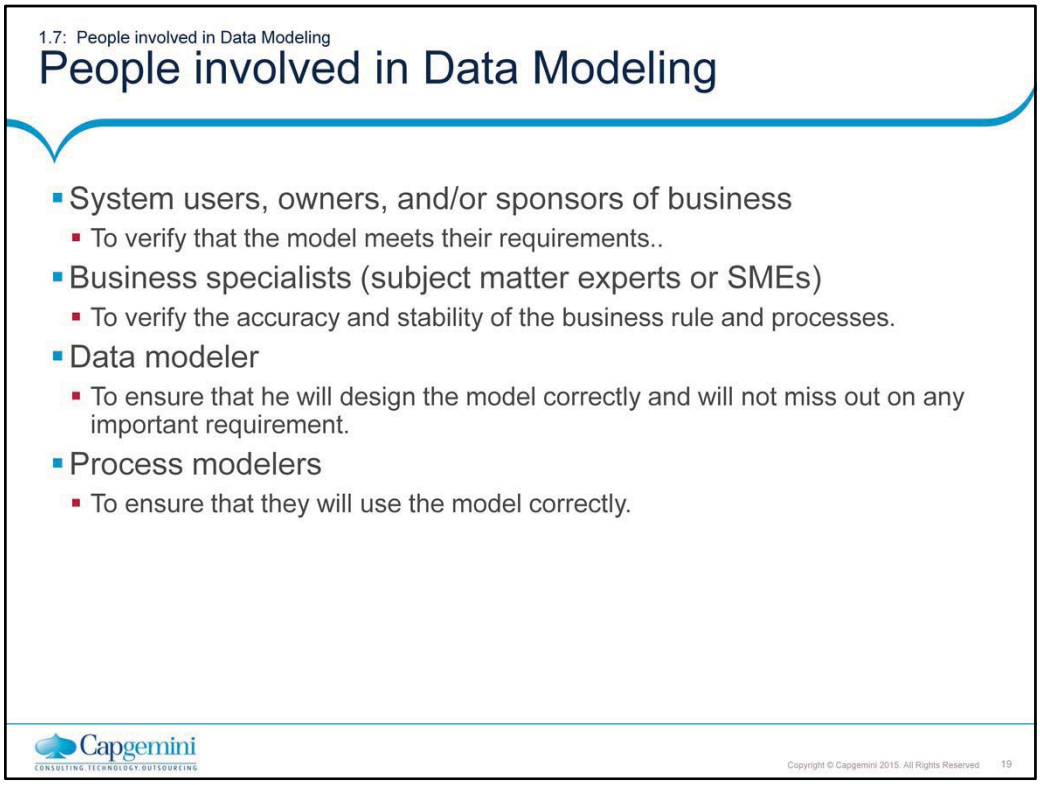
 17

What Data Modeling is not…

- A waste of time!
- A one time effort
- The ultimate IT application development cure
- A quick process
- A function solely performed and understood by and for IT professionals

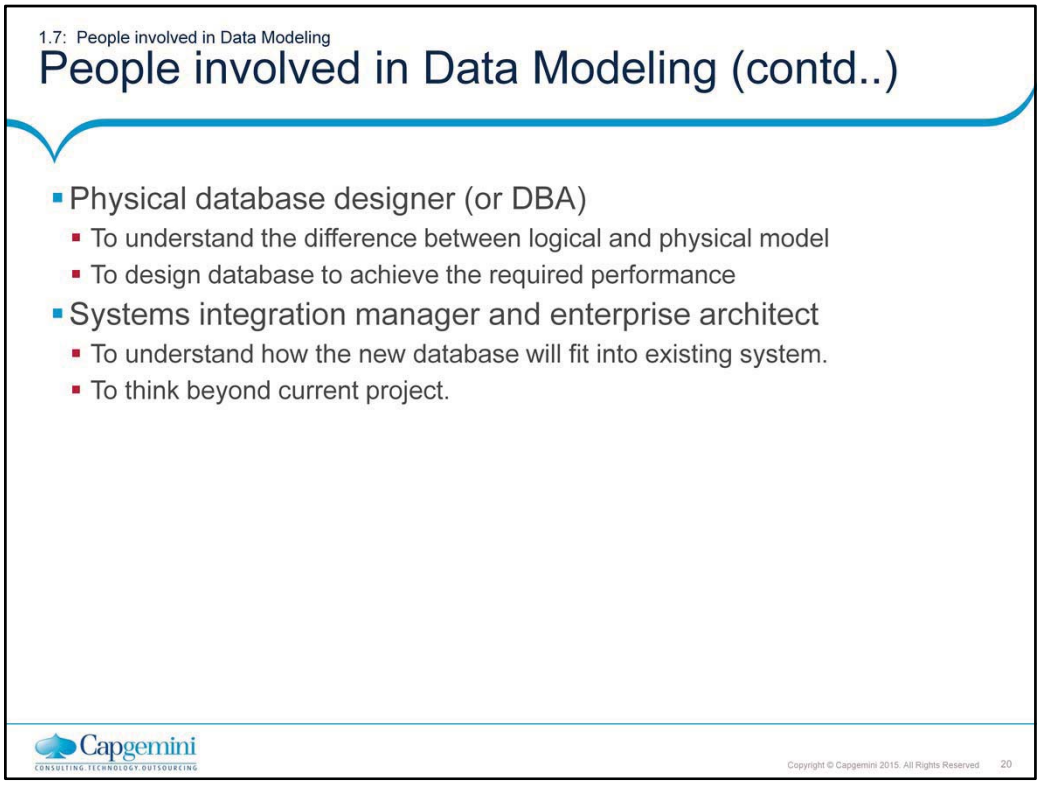
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18



Who should be involved in data modeling?:

•**System users, owners, and/or sponsors of business:** The system users, owners, and/or sponsors need to verify that the model meets their requirements. Our ultimate aim is to produce a model as the most cost-effective solution for the business. The users' *informed* agreement is an important measure taken towards achieving this aim.
•**Business specialists:** Business specialists (sometimes called Subject Matter Experts or SMEs) may be called upon to verify the accuracy and stability of business rules incorporated in the model. They themselves may not have any immediate interest in the system. For example, we might involve strategic planners to assess the likelihood of various changes to the organization's product range.
•**Data Modeler**: The data modeler has overall responsibility for developing the model and ensuring that other stakeholders are fully aware of its implications for them: "Do you realize that any change to your rule that each policy is associated with only one customer will be very expensive to implement later?"
•**Process modelers:** Process modelers and program designers need to specify programs to run against the database. They want to verify that the data model supports all the required processes without requiring unnecessarily complex or sophisticated programming. In doing so, they need to gain an understanding of the model to ensure that they use it correctly.

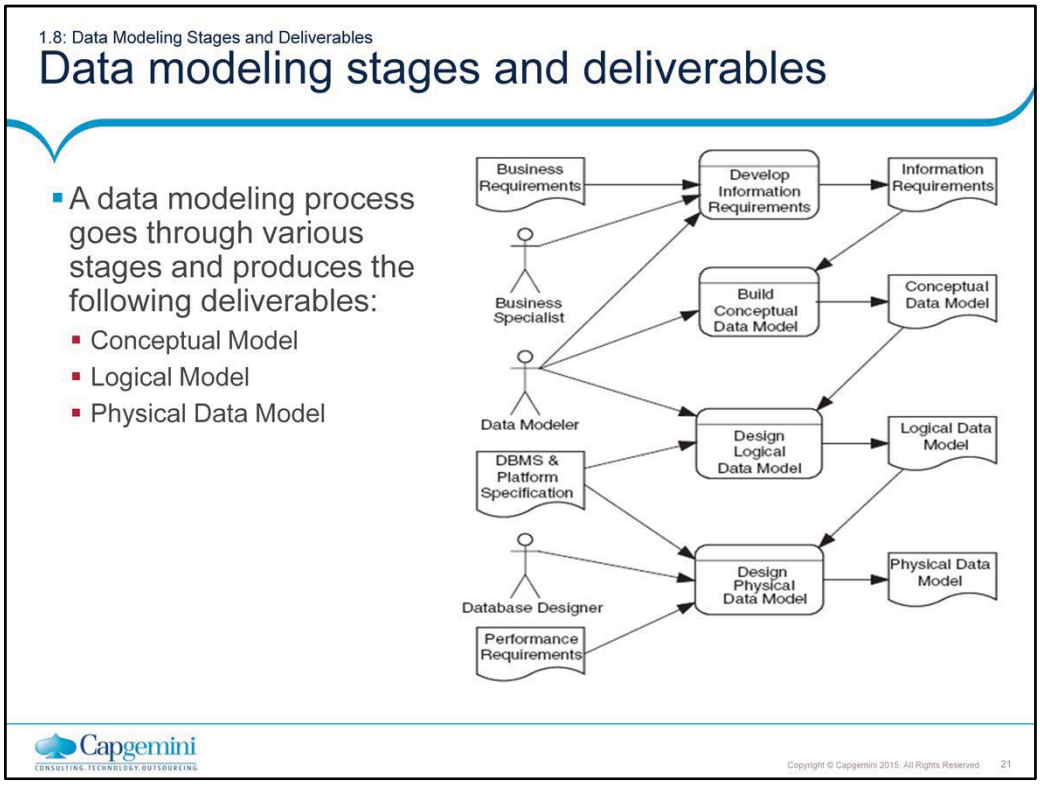


Who should be involved in data modeling? (contd.):

•**Physical Database Designer:** The physical database designer (often an additional role given to the database administrator) will need to assess whether the physical data model needs to differ substantially from the logical data model to achieve adequate performance, and, if so, propose and negotiate such changes. This person (or persons) will need to have an in-depth knowledge of the capabilities of the chosen DBMS.

•**Systems Integration Manager:** The systems integration manager (or other person with that responsibility, possibly an enterprise architect, data administrator, information systems planner, or chief information officer) is interested in how the new database fits into the bigger picture:

- ➢Are there any overlaps with other databases?
- ➢Does the coding of data follow organizational or external standards?
- ➢Have other users of the data been considered?
- ➢Are names and documentation in line with standards?
- ➢In encouraging consistency, sharing, and reuse of data, the integration manager represents business needs beyond the immediate project.



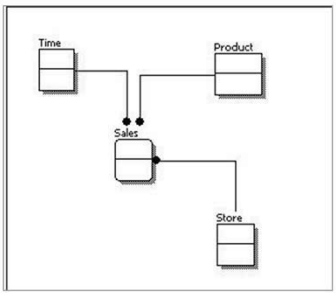
What are data modeling stages and deliverables?

- **Conceptual Model:** The conceptual data model is a (relatively) technology-independent specification of the data to be deposited and maintained in the database. The conceptual model is the focus of communication between the data modeler and business stakeholders, and it is usually presented as a diagram with supporting documentation.
- **Logical Model**: The logical data model is a translation of the conceptual model into structures that can be implemented using a Database Management System (DBMS). Today, that usually means that this model specifies tables and columns, as we saw in our first example. These are the basic building blocks of relational databases, which are implemented using a Relational Database Management System (RDBMS).
- **Physical Data Model**: The physical data model incorporates any changes necessary to achieve adequate performance and is also presented in terms of tables and columns, together with a specification of physical storage (which may include data distribution) and access mechanisms.

1.8: Data Modeling Stages and Deliverables

Conceptual Data Model

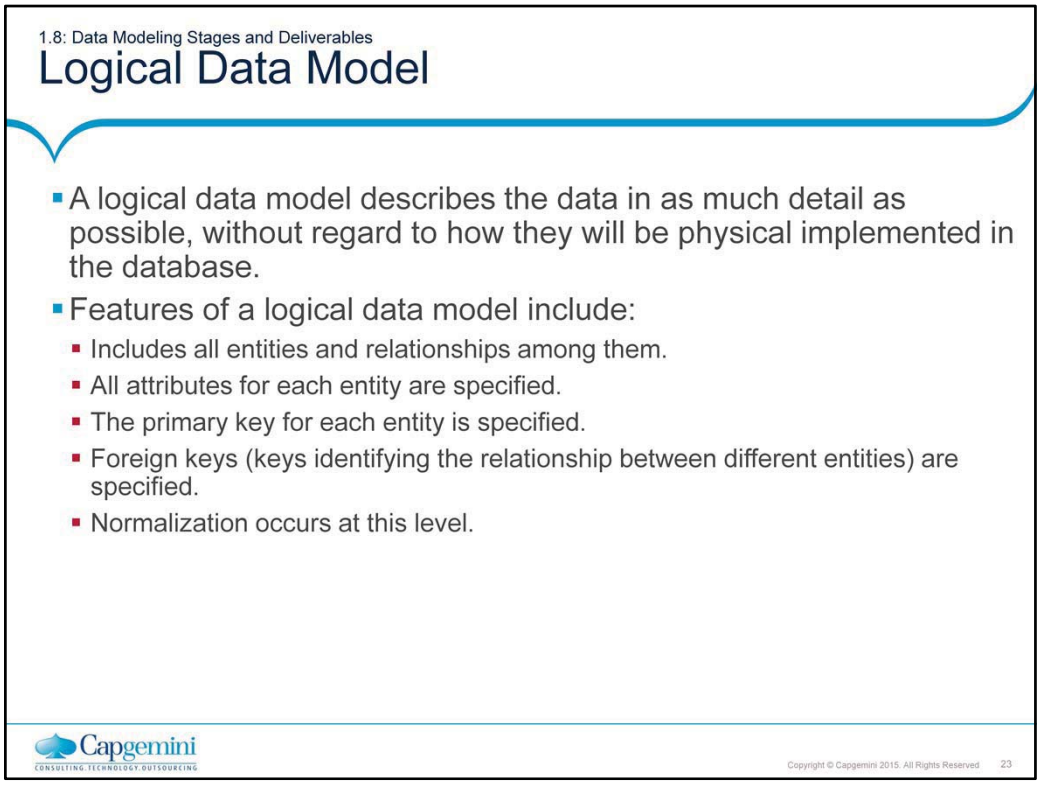
- A conceptual data model identifies the highest-level relationships between the different entities
  - Features of conceptual data model include:
  - Includes the important entities and the relationships among them.
  - No attribute is specified.
  - No primary key is specified



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

From the figure above, we can see that the only information shown via the conceptual data model is the entities that describe the data and the relationships between those entities. No other information is shown through the conceptual data model.



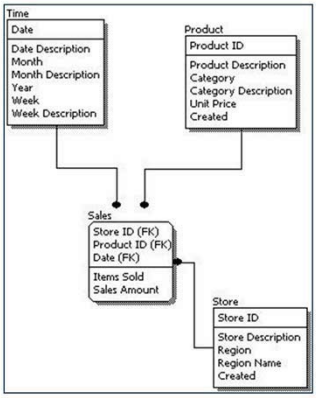
Comparing the logical data model shown above with the conceptual data model diagram, we see the main differences between the two:

- In a logical data model, primary keys are present, whereas in a conceptual data model, no primary key is present.
- In a logical data model, all attributes are specified within an entity. No attributes are specified in a conceptual data model.
- Relationships between entities are specified using primary keys and foreign keys in a logical data model. In a conceptual data model, the relationships are simply stated, not specified, so we simply know that two entities are related, but we do not specify what attributes are used for this relationship.

1.8: Data Modeling Stages and Deliverables

Logical Data Model (contd..)

- The steps for designing the logical data model are as follows:
  - Specify primary keys for all entities.
  - Find the relationships between different entities.
  - Find all attributes for each entity.
  - Resolve many-to-many relationships.
  - Normalization.



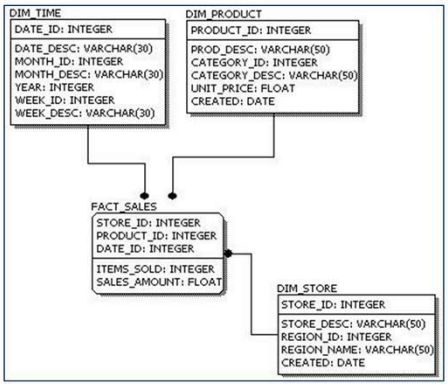
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24

1.8: Data Modeling Stages and Deliverables

Physical Data Model (contd..)

- Physical data model represents how the model will be built in the database.
- A physical database model shows all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships between tables.



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25

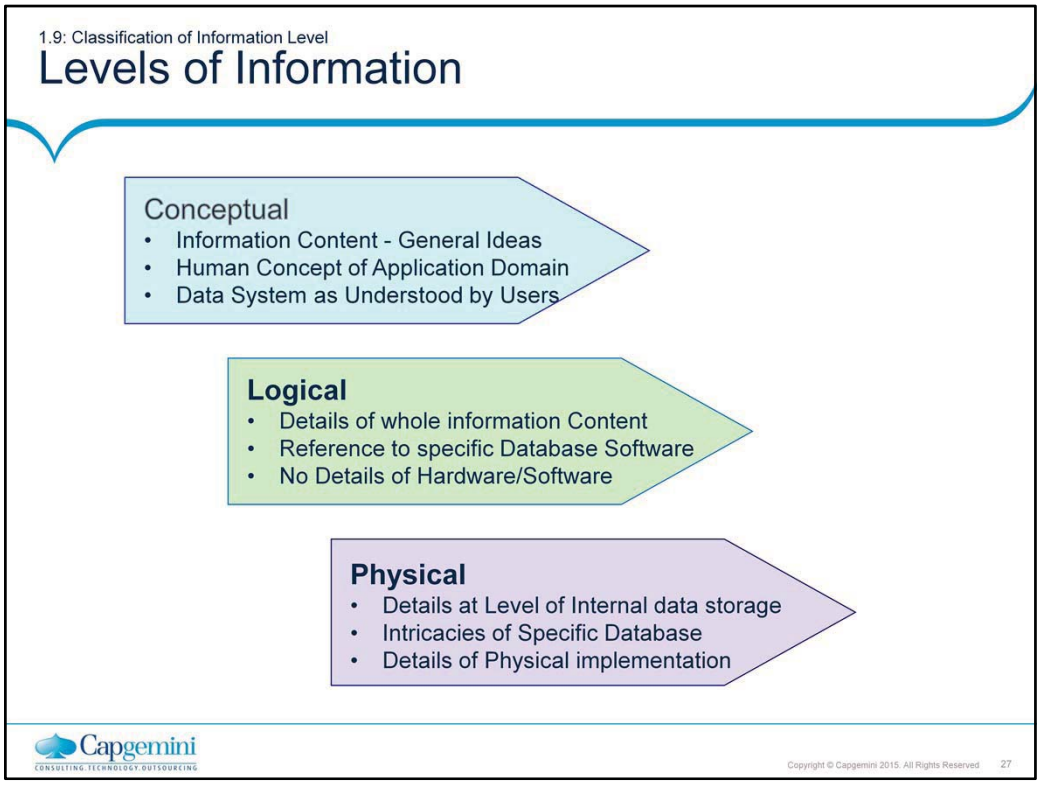
1.8: Data Modeling Stages and Deliverables

Physical Data Model (contd..)

- Features of a physical data model include:
  - Specification of all tables and columns.
  - Foreign keys are used to identify relationships between tables.
  - Demoralization may occur based on user requirements.
  - Physical considerations 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 between MySQL and SQL Server

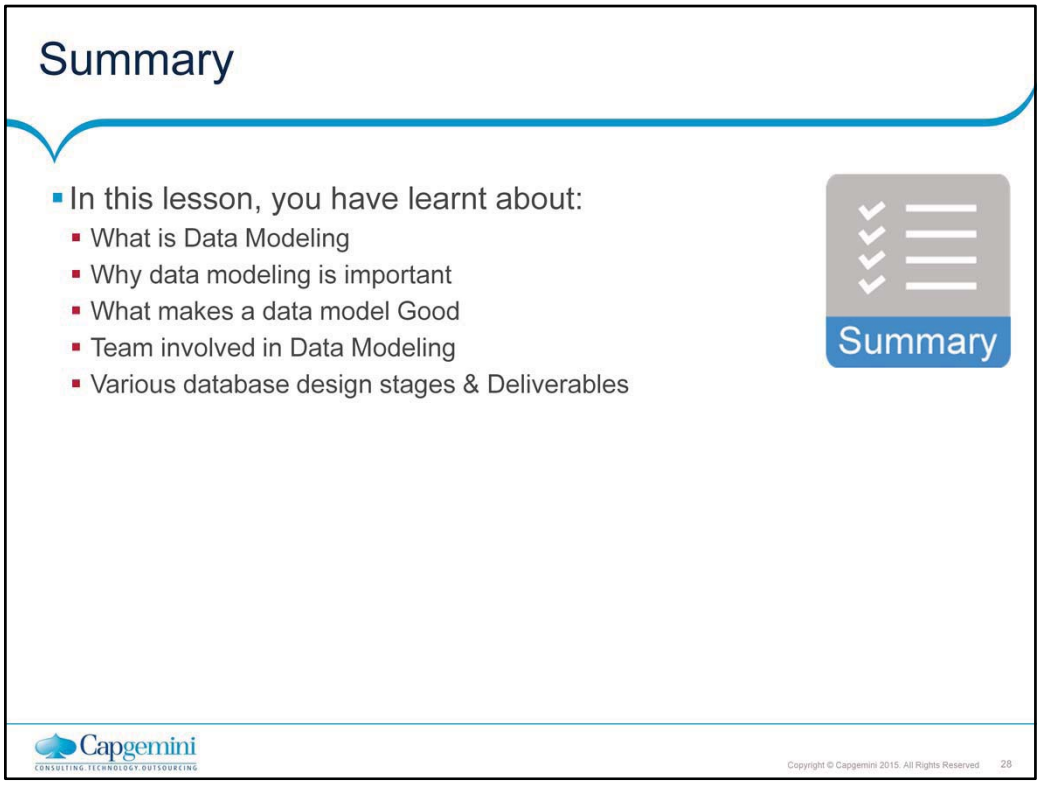
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26



Levels of Information:

- **Conceptual Level:** This is the highest level consisting of general ideas about the information content. At the conceptual level, the data model represents the information requirements of the entire set of user groups in the organization. At this level, you have the description of application domain in terms of human concepts. This is the level at which the users are able to understand the data system. This is a comprehensive, complete and stable information level.
- **Logical Level:** At this level, the domain concepts and their relationships are explored further. This level accommodates more details about the information content. Still, storage and physical considerations are not part of this level. Considerations of a specific DBMS **may not** find a place at this level.
- **Internal or Physical Level:** This information level deals with the implementation of the database on secondary storage. Considerations of storage management, access management, and database performance apply at this level. Here intricacy and complex details of the particular database are relevant. The intricacies of the particular DBMS are taken into account at the physical level.



Add the notes here.

Review Question

- Question 1: _____ is a replica or a representation of particular aspects and segments of the real world.
- Question 2: _____ data model represents how the model will be built in the database.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 29

Data Modeling for Business Intelligence

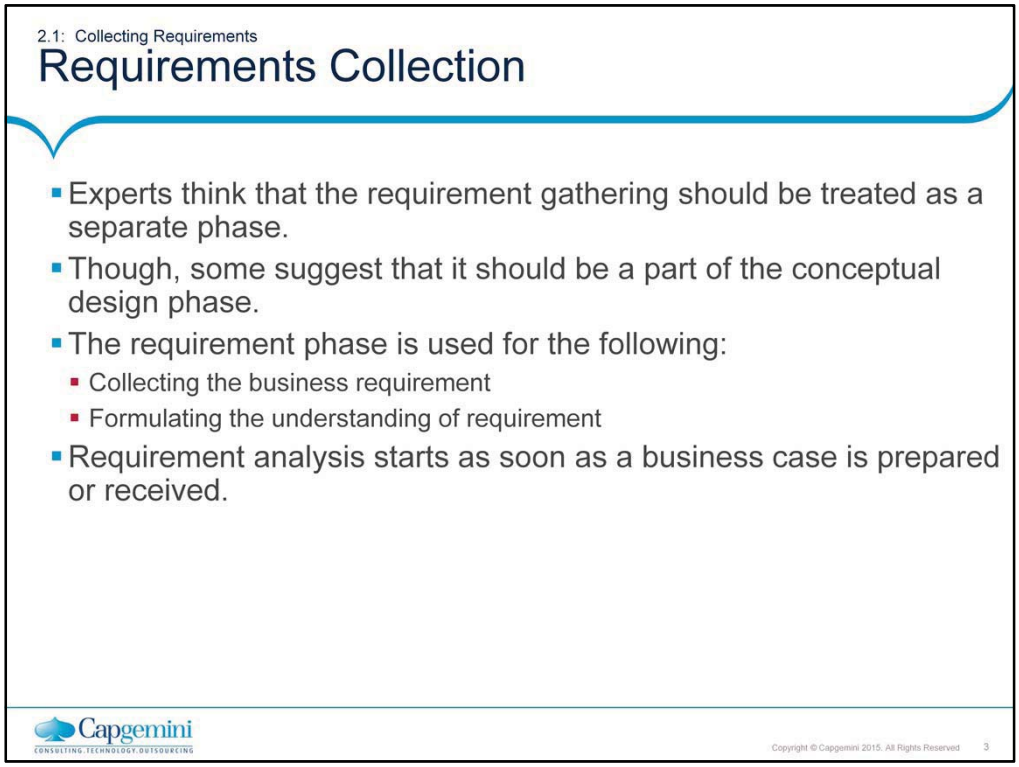
Lesson 2: Understanding Business Requirements

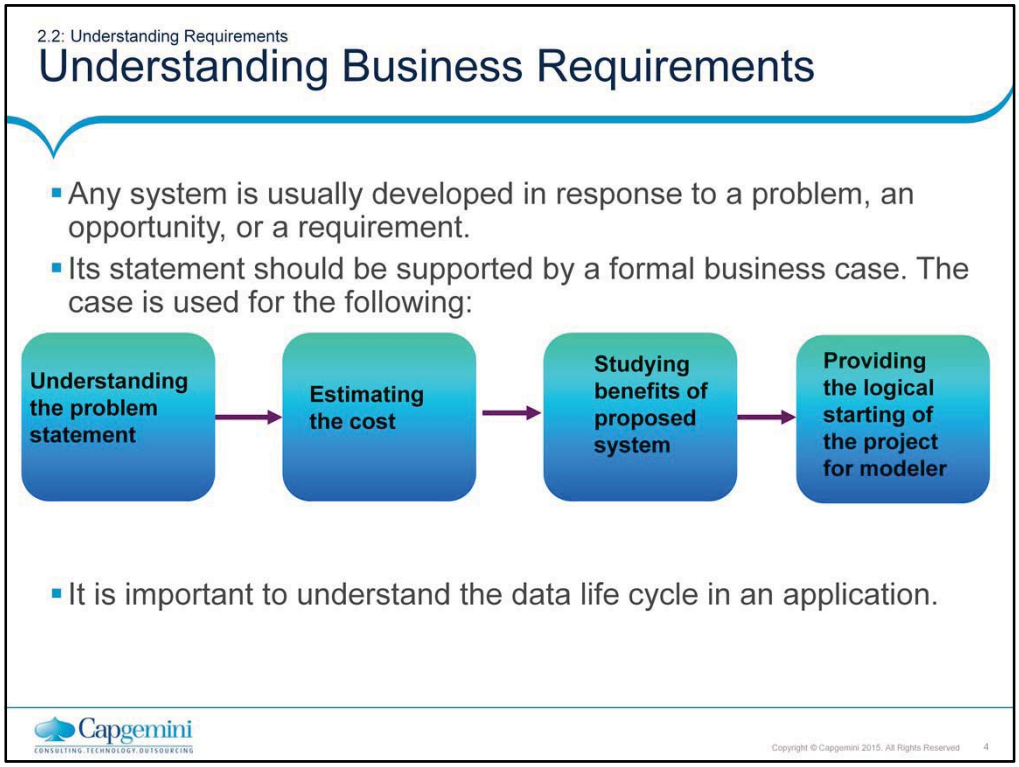
Lesson Objectives

- This lesson will provide overview of various techniques of Requirement gathering
- We will learn about:
  - Need of Requirement Analysis
  - The Data Life cycle
  - Ways of Collecting requirement
  - Business Requirement Specification (BRS)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

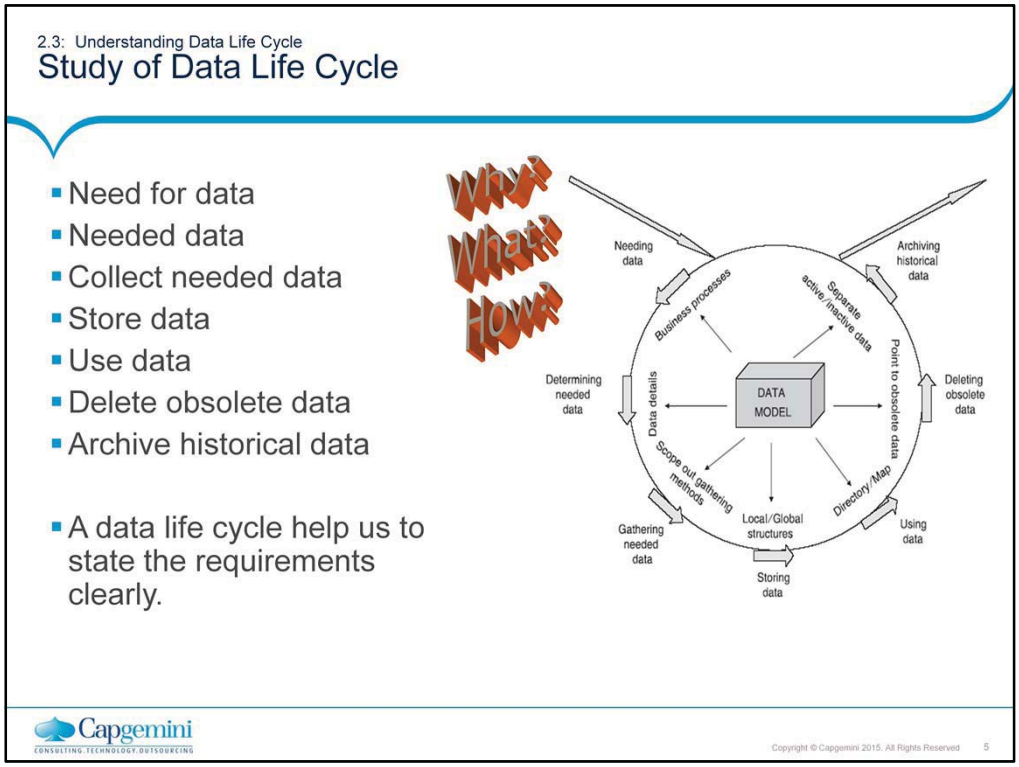

2.1: Collecting Requirements
Requirements Collection
Experts think that the requirement gathering should be treated as a separate phase.
Though, some suggest that it should be a part of the conceptual design phase.
The requirement phase is used for the following:
Collecting the business requirement
Formulating the understanding of requirement
Requirement analysis starts as soon as a business case is prepared or received.
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
3



Understanding Business Requirements:
Typically, the business case:
Estimates the costs and benefits
Studies the risks of alternative approaches
Recommends a particular direction
Provides the logical starting point for the modeler in understanding the context and requirements

While understanding the business case, one should specify the following very carefully:
A detailed justification of the application. Who will benefit from it? Does it have any disadvantages?
The business concepts, rule and terminology
The critical success factor to the application
The scope of the system
System size and time requirement
Performance related requirements, if any
Expected life of application
Connectivity or interface with other existing or expected applications

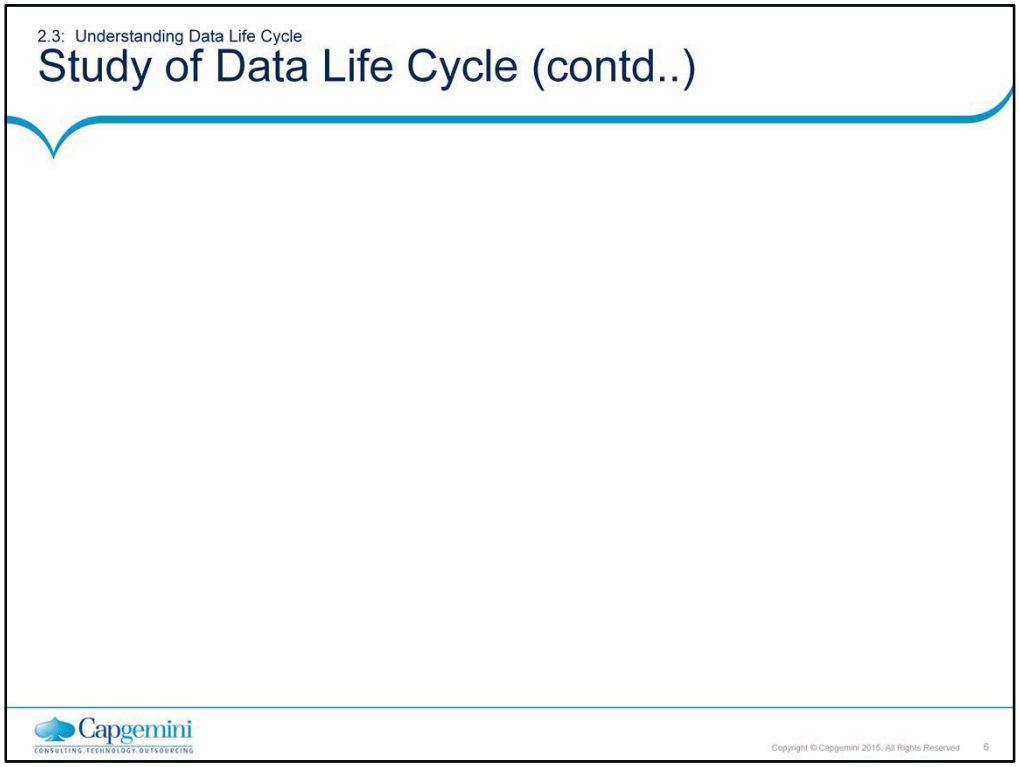
While understanding the requirements, a modeler needs to study the overall data life cycle and how to use the data at various stages.



Study of Data Life Cycle:

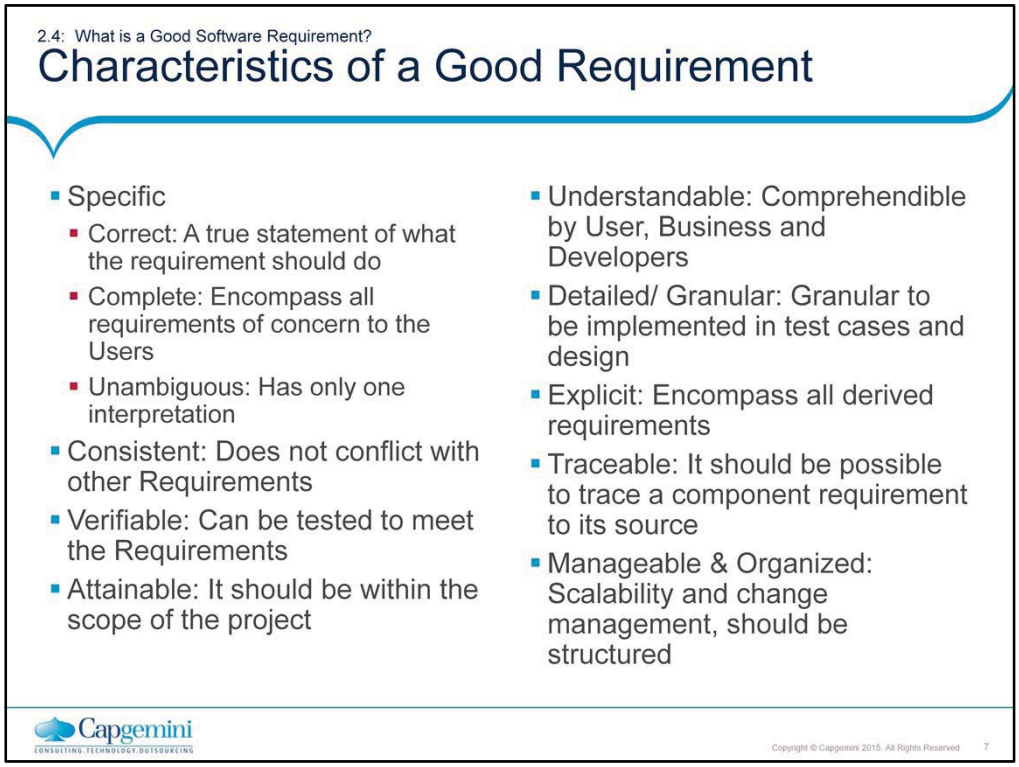
Need for Data: After the business process or a problem is identified, you need to understand the various data-related needs of the business process. The data passes through various stages and it is important to decide how to use it in every stage.
Needed Data: It is used to clearly record what data is needed to run the application. What needs to be recorded and what should be discarded. All the required details of the needed data elements are discovered and documented in the data model.
Collect Needed Data: After identifying which data is needed, collection of data takes place. Here, you apply a sort of filter to gather only the data that is needed and ignore the irrelevant data that is not necessary for any of your business processes. One needs to clearly define the various methods of collecting data.
Store Data: The collected data must be stored in the database using appropriate methods of storage. You need to decide and include the storage medium after considering the optimal storage method to suit the needs of users for accessing and using data.
Use Data: That is the ultimate goal in the data life cycle. At this stage, you perform the following activities:

- Combine various data elements
- Retrieve data elements for usage
- Modify and store modified data
- Add new data created during the business processes



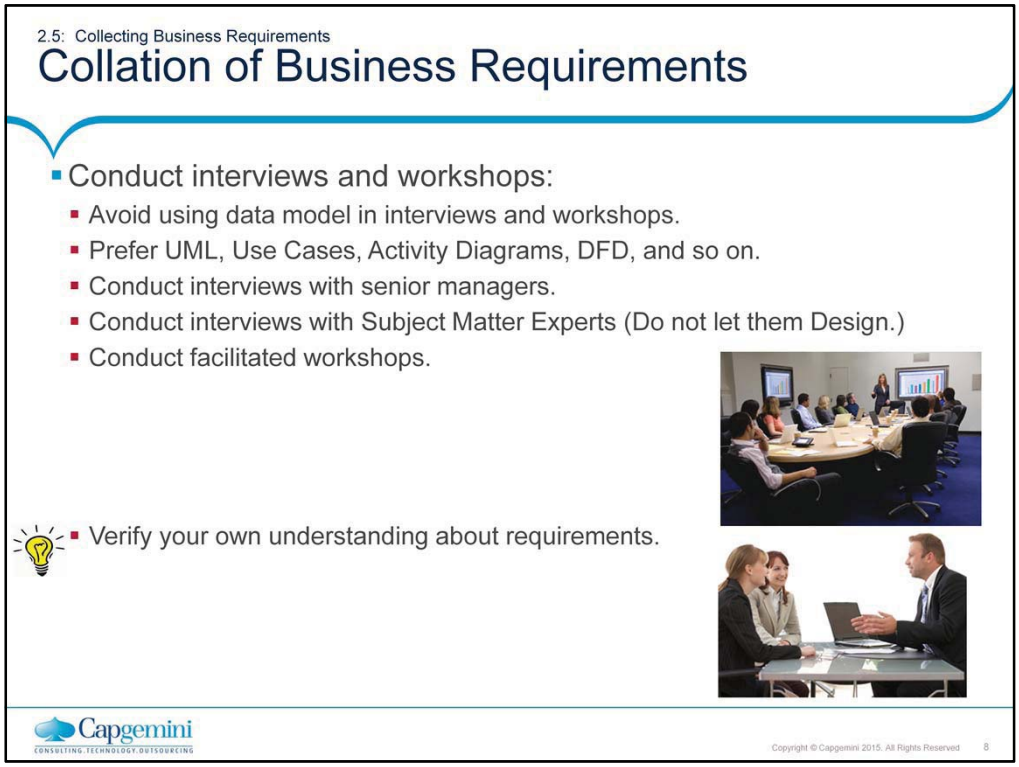
Study of Data Life Cycle (contd.):

Delete Obsolete Data: After a particular time period, a particular data element in storage may become old and obsolete. After a period of time, the data element may no longer be useful and, therefore, not accessed in any transactions at all. Presence of such data in the system will slowdown the performance and introduce maintenance-related issues. Deleting such obsolete data becomes an ongoing operation. A particular data element may fall into the category qualified for deletion. At this stage, the data model is used to examine the various data elements that can be safely deleted after specified periods.
Archive Historical Data: Some data elements could be useful even after any activity on those data elements had ceased. Historical data can be used in the data warehouse of the organization. Any such useful data elements are removed from the current database and archived into a separate historical repository for further use.

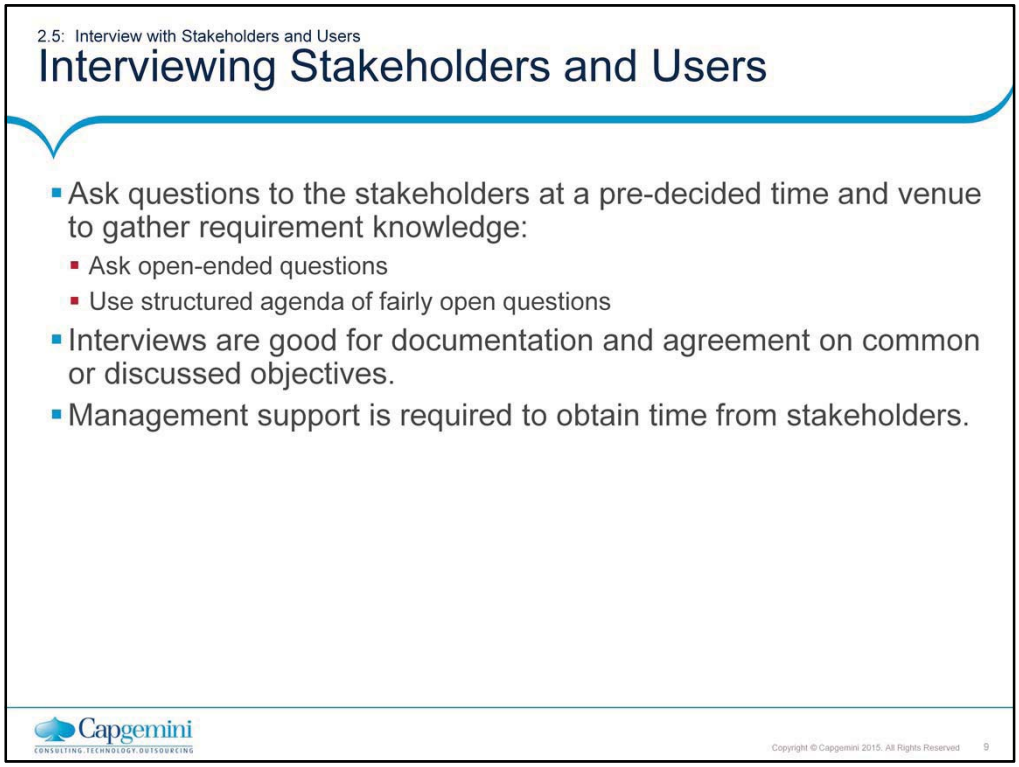


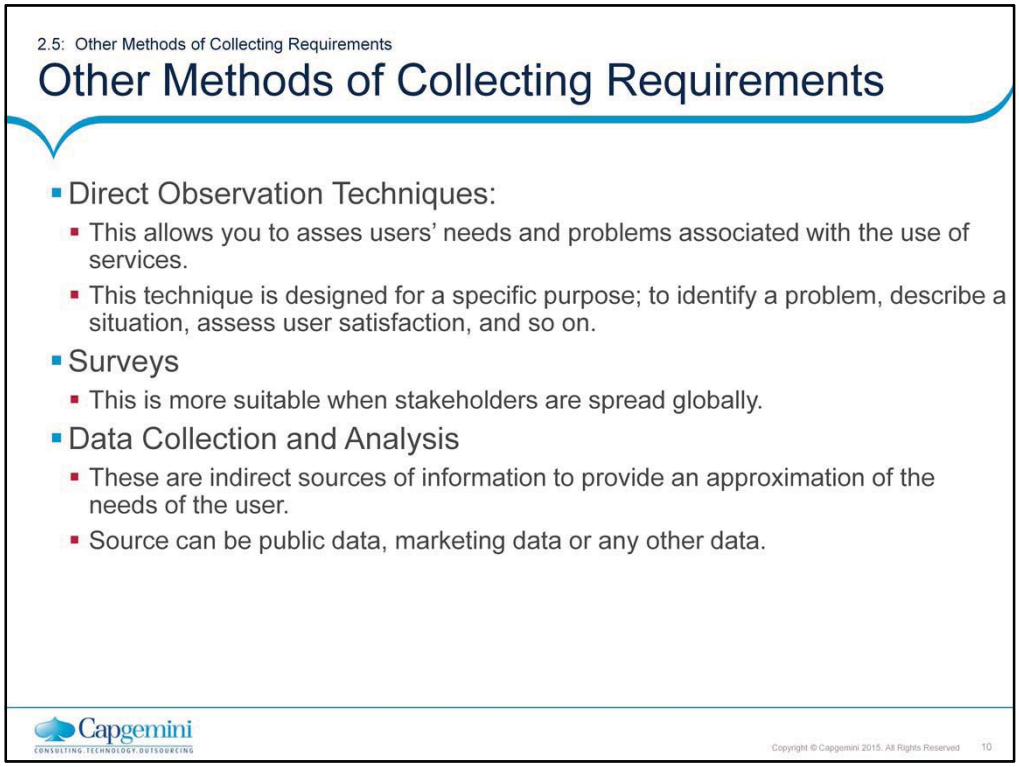
Characteristics of a Good Requirement:

Meeting the customer's real needs is one of the goals of developing a system. If the system does not meet the customer's needs, then the perceived value of the system diminishes.



Collection of Business Requirements:
Interviews and workshops are the essential techniques for requirements gathering.
You need to be very careful about using data models as your means of communication during these initial interviews or workshops. In fact, use anything but data models: UML Use Cases and Activity Diagrams, plain text, data flow diagrams, event diagrams, function hierarchies, and/or report layouts.
CEOs and other senior managers may not be familiar with the details of process and data but are usually the best placed to paint a picture of future directions.
Business experts, end users, and “subject matter experts” are the people we consult in order to understand the data requirements in depth. Do not let them design the model—at least not yet! Instead, encourage them to explain the processes and the data they use and to examine critically how well their needs are met.


2.5: Interview with Stakeholders and Users
Interviewing Stakeholders and Users
Ask questions to the stakeholders at a pre-decided time and venue to gather requirement knowledge:
Ask open-ended questions
Use structured agenda of fairly open questions
Interviews are good for documentation and agreement on common or discussed objectives.
Management support is required to obtain time from stakeholders.
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
9

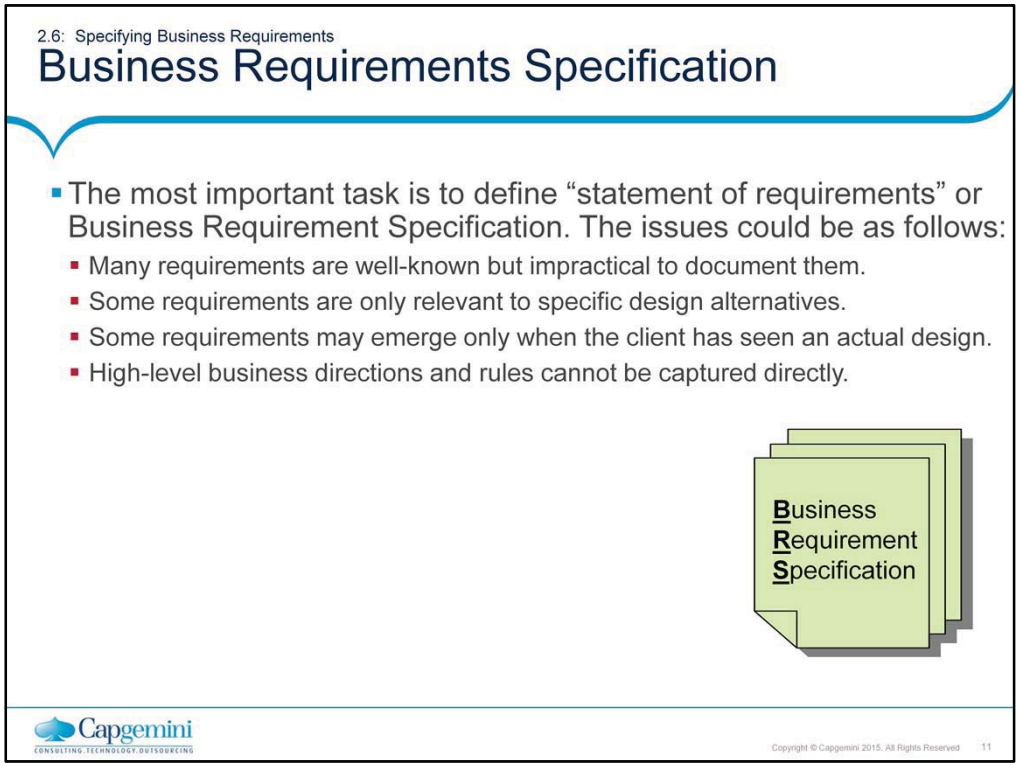


Other Methods of Collecting Requirements:

Direct Observation Techniques
It reveals details which other methods cannot.
It also has limitations such as the following:
a) Extremely time consuming
b) Expensive and requires careful observation
c) Results maybe hard to analyze as it yields too much data

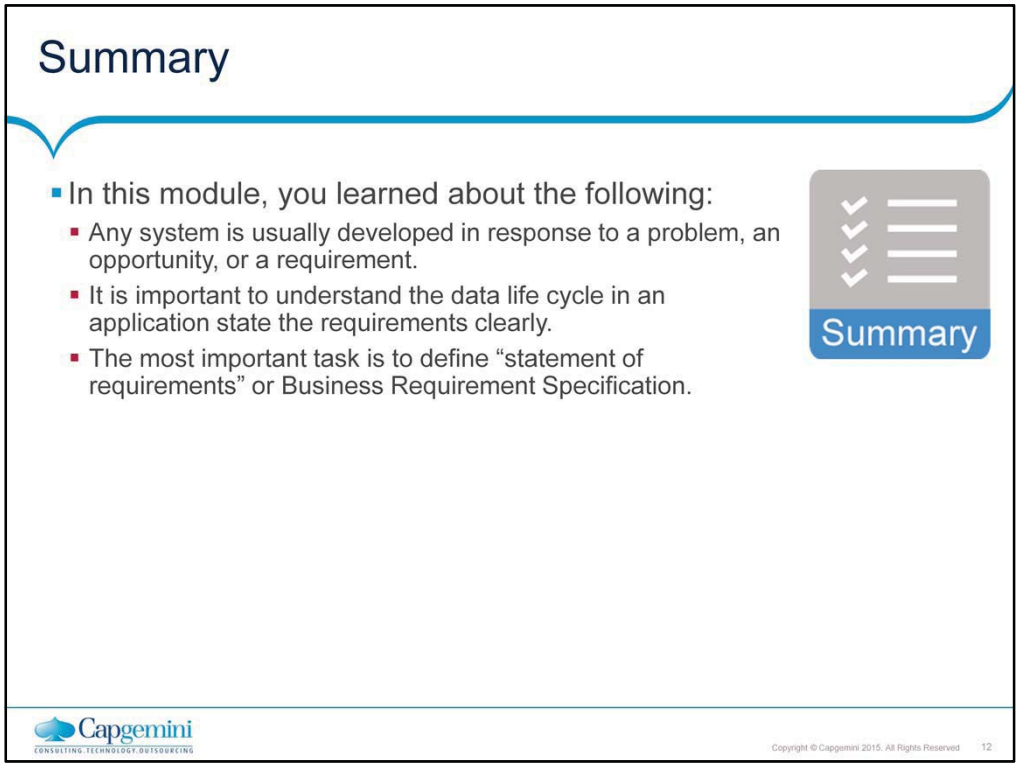
Surveys
Surveys are more suitable when stakeholders are spread across locations.
They are used to collect information from many users in less time.

Data Collection and Analysis
Indirect sources of information to provide an approximation of the needs of the user
Source can be public data, marketing data, or any other sources.



Business Requirements Specification:

Some of the issues with the preparation of BRS document:
Many requirements are well-known to the designer and client (“The house must be structurally sound; the shower requires both hot and cold water.”) and it would be impractical to try to document them in full.
Some requirements are only relevant to specific design alternatives (“The shelves in this cupboard should be widely spaced,” only makes sense in the context of a design that includes the cupboard).
Some requirements may emerge only when the client has seen an actual design (“I like to sleep in complete darkness.” or “I don’t want to hear the kids practicing piano.”).
high-level business directions and rules cannot be captured directly: “We need to be able to introduce new products without redesigning the system.”



Add the notes here.

Review Question

- Question 1: _____ is more suitable when stakeholders are spread globally.
- Question 2: _____ are good for documentation and agreement on common or discussed objectives

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

Data Modeling for Business Intelligence

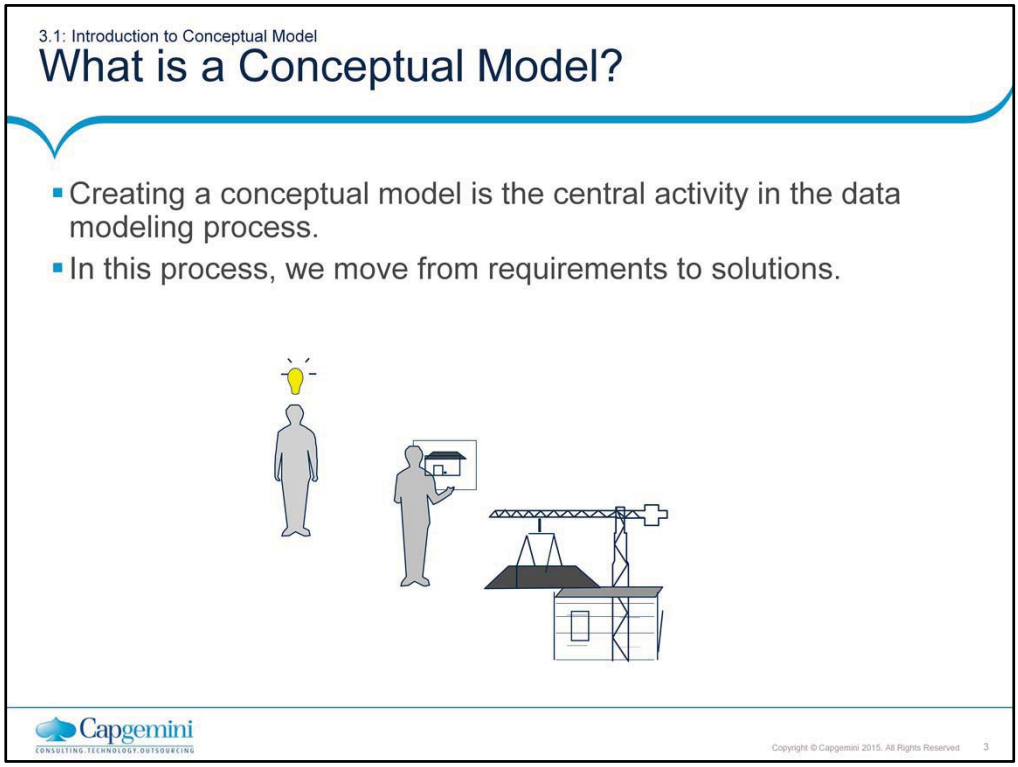
Lesson 3: Conceptual Model

Lesson Objectives

- On completion of this lesson, you will be able to:
  - Define conceptual model
  - State objectives of conceptual model and list its components
  - List and describe main stages in conceptual modeling
  - Describe Online Transaction Processing System
  - State advantages of using generic model
  - Describe the components of a generic model
  - Identify steps of dimension modeling

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



Conceptual data model includes all major entities and relationships and does not contain much detailed level of information about attributes and is often used in the INITIAL PLANNING PHASE. Conceptual data model is created by gathering business requirements from various sources like business documents, discussion with functional teams, business analysts, smart management experts and end users who do the reporting on the database.

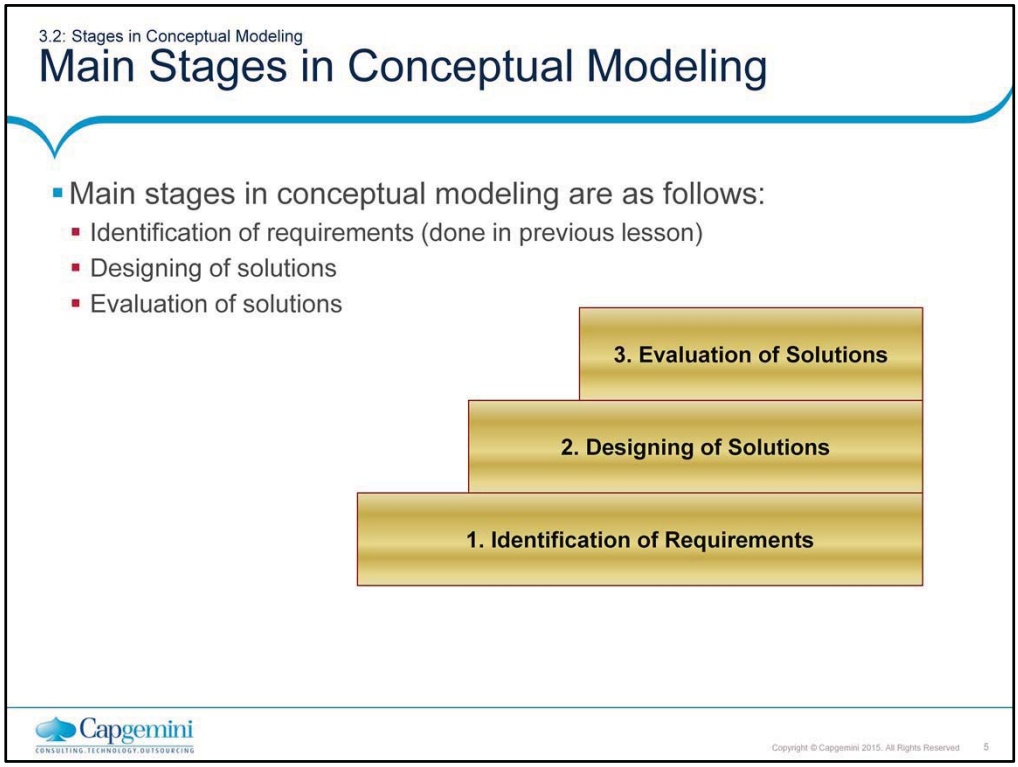
3.1: Goals of Conceptual Model

Objectives of a Conceptual Model

- All pieces of information that are required to run a business are properly recognized.
- Every single piece of required information is displayed only once in the model.
- The main consideration is, in the future system, the information should be available in a predictable and logical place.
- Related information is kept together.
- A proper Entity-Relationship (ER) model leads to a set of logically coherent tables.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

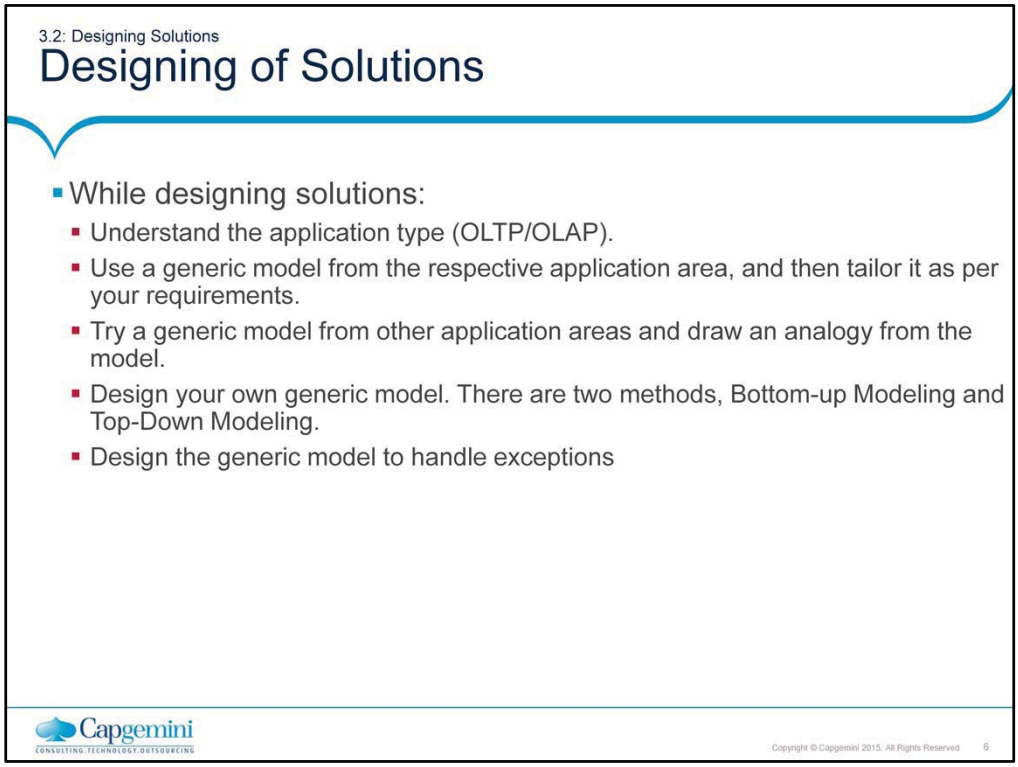
 4



Main Stages in Conceptual Modeling:

Conceptual modeling has various stages, starting from identification of requirements to design of solutions and finally to their evaluation. All these stages provide basic inputs to conceptual modeling process, which gets fine-tuned in the later stages of modeling.

Refer to lesson 2 for Identification of Requirements.



Designing of Solutions:

While designing, usually try to find a generic model that broadly meets the users' requirements, and then tailor it to suit a particular application, drawing on standard structures and adapting structures from other models as opportunities arise.

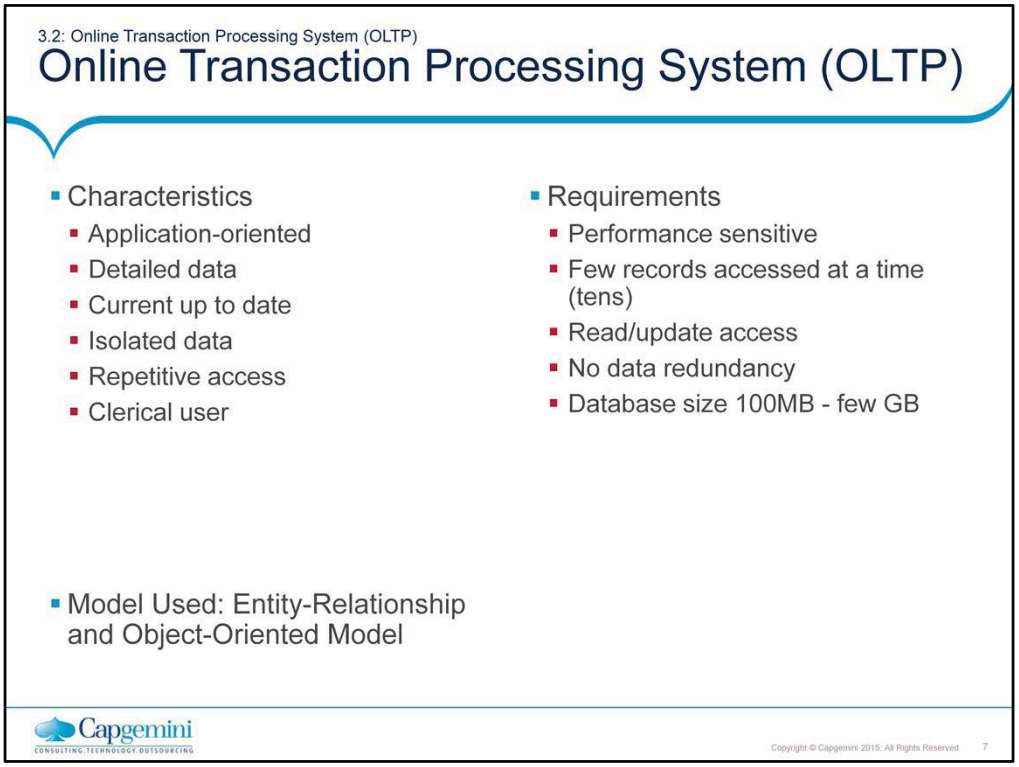
Sometimes, you may not have an explicit generic model available, however, you can draw an analogy with a model from a different field. Try using Life Insurance model for Health Insurance System.

There are two methods for designing a generic model, Bottom-up Modeling and Top-Down Modeling.

The Bottom-up approach: You initially develop a very "literal" model, based on existing data structures and terminology. Then, you use subtyping and super typing to move towards other options. You need not be creative; however, the model should be improvable over a period of time.

The Top-Down approach: We simply use a model that is generic enough to cover at least the main entity classes in any business or organization.

While designing model, try to make is flexible. Add necessary structures required to handle it. Try to optimize the common situations to handle it.


3.2: Online Transaction Processing System (OLTP)
Online Transaction Processing System (OLTP)
Characteristics
Application-oriented
Detailed data
Current up to date
Isolated data
Repetitive access
Clerical user
Requirements
Performance sensitive
Few records accessed at a time (tens)
Read/update access
No data redundancy
Database size 100MB - few GB
Model Used: Entity-Relationship and Object-Oriented Model
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
7

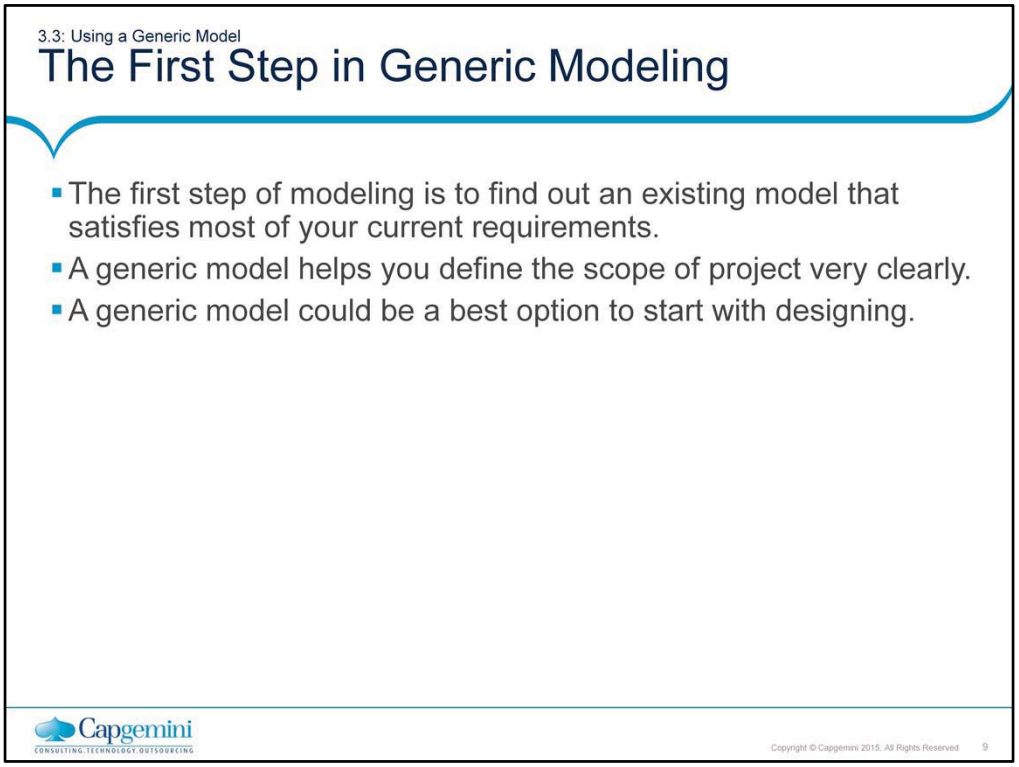
3.3: Patterns and Generic Models

Advantages of Patterns and Generic Models

- No one likes to start the modeling from scratch.
- Modeler very much relies on proven/used structures.
- The advantages of using an existing model or a generic model are as follows:
  - It helps us to understand the system better.
  - It saves a lot of development time and efforts.
  - For example, use of life insurance model could be very useful for designing a model for health insurance.
  - It is known to the modeler.

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 8



For example, we may need to develop a data model to support human resource management. Suppose we have seen successful human resources models in the past, and have (explicitly or just mentally) generalized these to produce a generic model.

3.3: Adopting Generic Model from Other Applications

Reusing Generic Models of Other Applications

- In absence of a generic model in a required application, a generic model from other application can be used.
- Consider that you need to build a property insurance model and you don't have any model readily available. In such case, a model from life insurance or health insurance could be used.

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 10

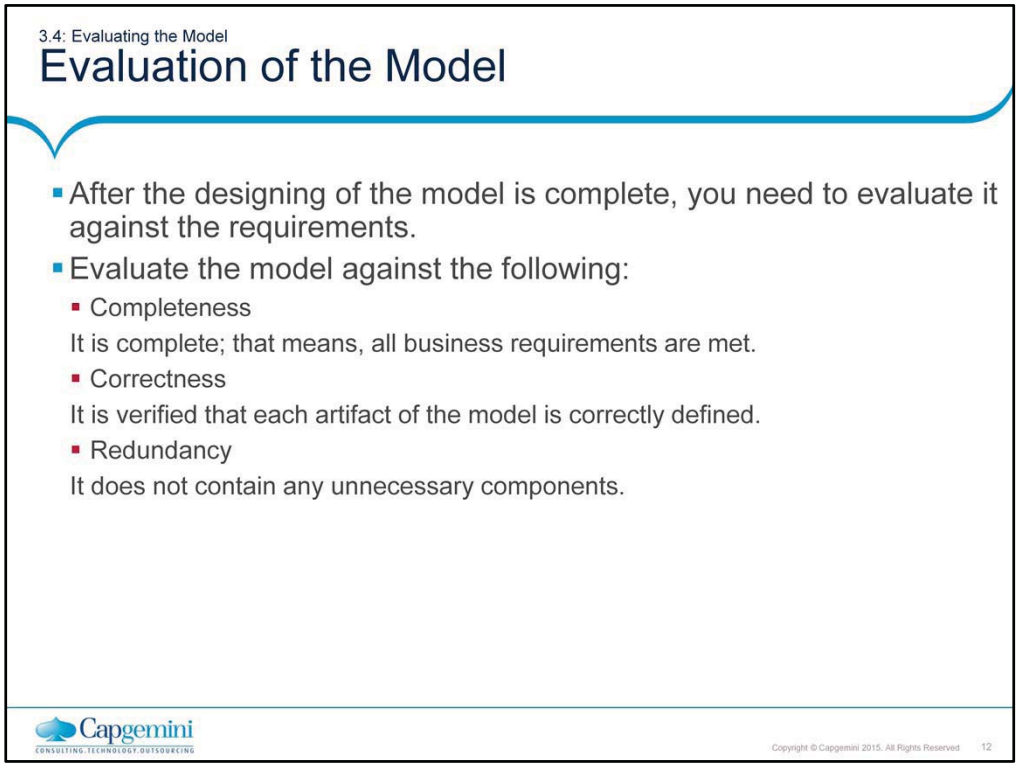
3.3: Generic Model

When there is no generic Model

- In case no generic model is available, a new model should be developed.

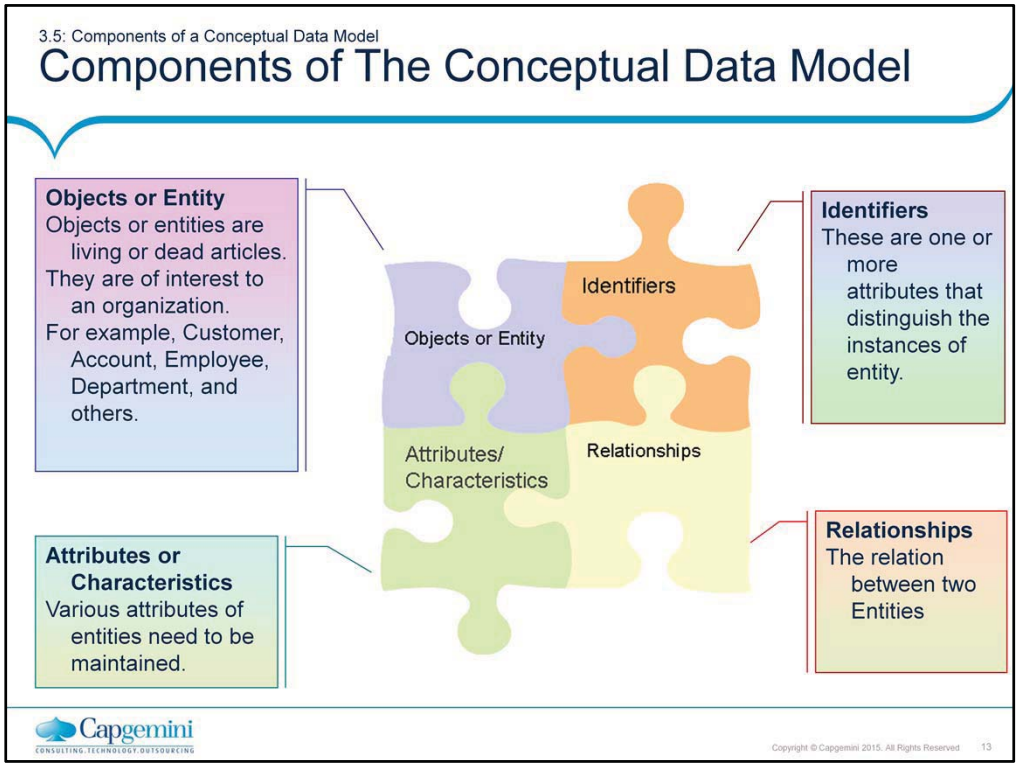
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11



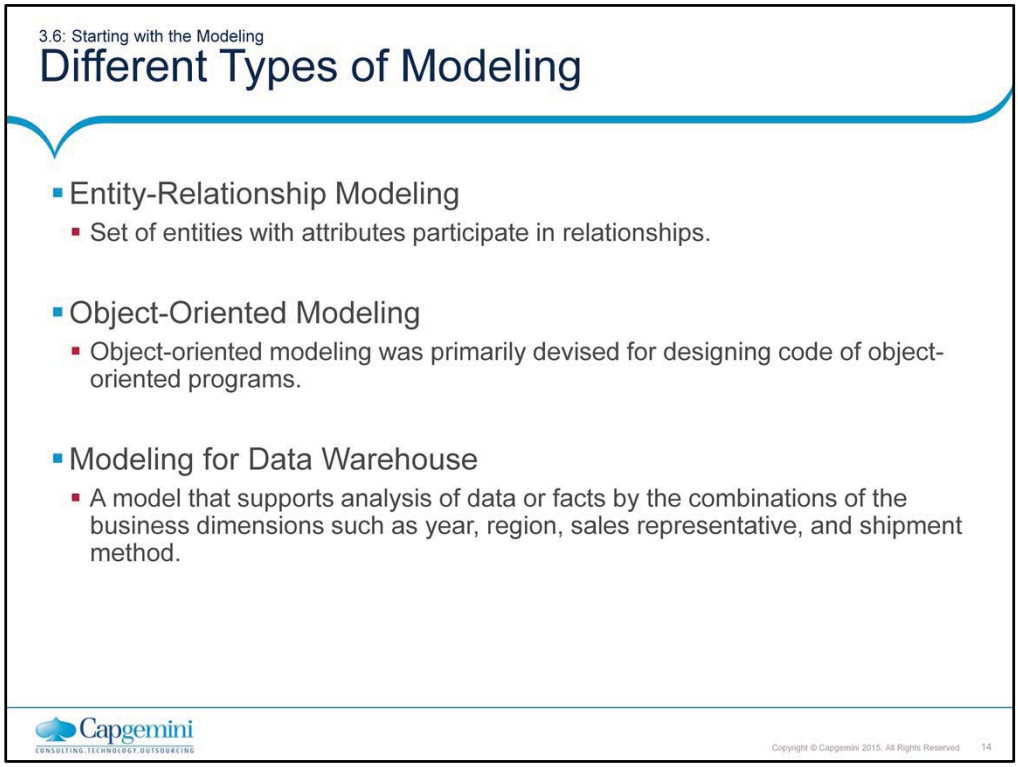
Evaluation of the Model:

Having developed one or more candidate conceptual models, you need to select the most appropriate alternative and verify that it meets the business needs. Perform the evaluation thoroughly at this step. You will then require to only review the design decisions that you make as you proceed from the conceptual to logical and to physical models, rather than reviewing the later models in their entirety.



Components of The Conceptual Data Model:

- **Objects or Entity:**
  When you analyze the information requirements of a company, you will notice that the company needs information about the business objects significant for it.
- **Attributes:**
  Each customer has an intrinsic characteristic known as Customer Name. Every customer has a specific name. Every customer has other inherent or intrinsic characteristics such as Customer Address, Customer Phone Number, Customer Balance, and so on.



Different Types of Modeling:

Entity-Relationship Modeling:

This approach, introduced by Peter Chen in 1976, is still the most popular and widely-used technique. Vendors have produced several computer-aided software engineering (CASE) tools to support this method. This method perceives and portrays the information requirements of an organization as a set of entities with their attributes participating in relationships.

The ER model portrays the information domain of an organization in a way that it is free from any considerations of database software or hardware. Because of this independency, this method is well-suited for conceptual data modeling. It does not burden the domain experts with unnecessary details. However, an ER data model diagram has its shortcomings. The diagram does not clearly indicate constraints in the relationships.

Fact-Oriented Modeling

Not all domain experts are comfortable with the notations in the ER model. Some of them find some of the notations, especially those for relationships, incomplete and imprecise. The fact-oriented data modeling approach attempts to overcome some of the deficiencies of the ER approach.

In the 1970s, an approach to data modeling arose by viewing the information domain in terms of objects playing roles. A role is the part played by an object in a relationship. Object-role modeling (ORM) is such a fact-oriented modeling approach. This is perhaps the only major fact-oriented modeling technique with fairly wide industry support.

Compared with ORM, ER has the following shortcomings:

•It is not closer to natural language for validation by domain experts.

•ER techniques generally support only two-way relationships. N-way relationships in ER are broken down into two-way relationships by introducing intersection identities. However, these intersection identities seem arbitrary and not understood by domain experts.

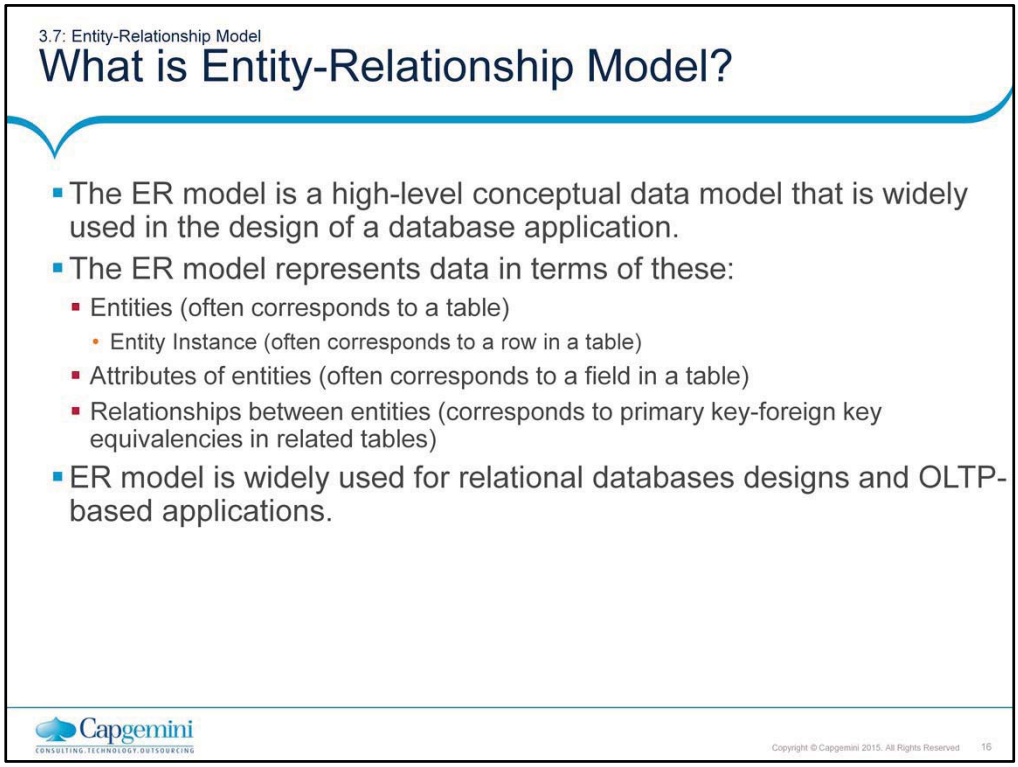
Object-Oriented Modeling: In this approach, both data and behavior are encapsulated within objects. Thus, object-oriented modeling was primarily devised for designing code of object-oriented programs. However, this modeling approach can be adapted for conceptual modeling and eventually for database design.

Till today, the most popular and widely used object-oriented approach is the Unified Modeling Language (UML). The Unified Modeling Language has an array of diagram types, and class diagrams form one important type. Class diagrams can represent data structures and may be considered as extensions of the ER technique.

Data Warehousing Model

As businesses grow, data management becomes more complex. Business executives desperately seek information to stay competitive, improve the bottom line and, importantly, to make strategic decisions. Companies accumulate vast quantities of data in their OLTP systems, but these systems themselves could not support intricate queries and analysis for providing strategic information.

A data warehouse must contain data extracted from OLTP systems — data that can be viewed and modeled for querying and analysis.



What is Entity-Relationship Model?

Entities are the principal data objects about which information is to be collected. Entities are usually recognizable concepts, either concrete or abstract, such as person, places, things, or events which have relevance to the database. Some specific examples of entities are EMPLOYEES, PROJECTS, INVOICES. An entity is analogous to a table in the relational model.

An *entity occurrence* (also called an instance) is an individual occurrence of an entity. An occurrence is analogous to a row in the relational table.

Attributes

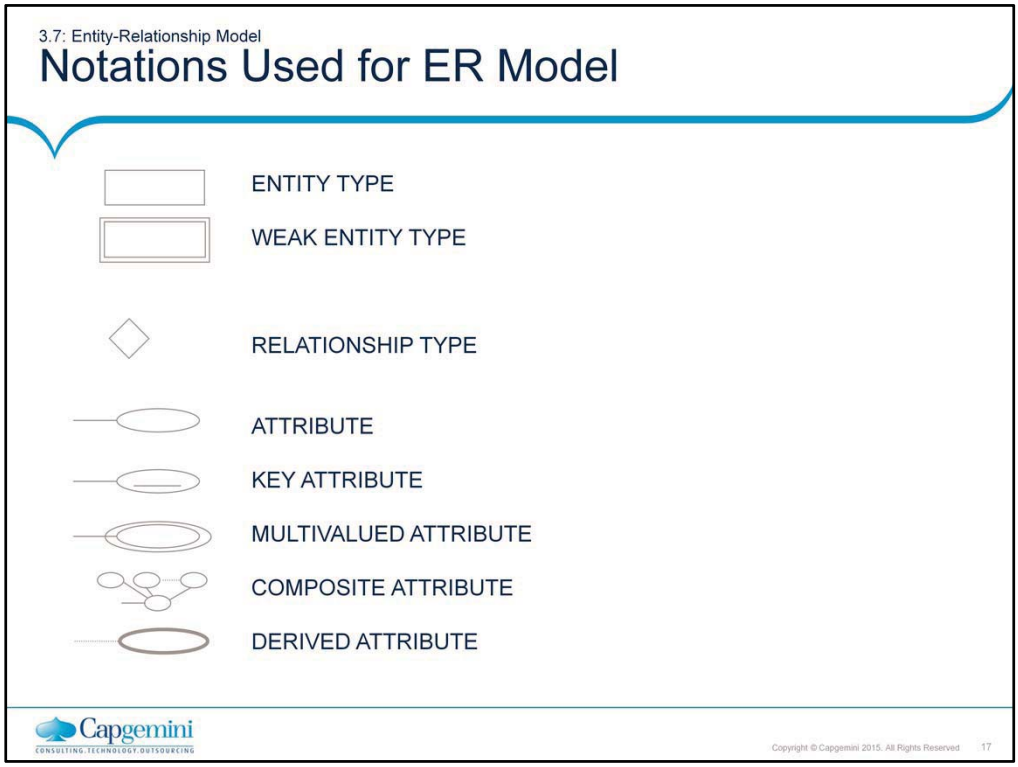
Attributes describe the entity with which they are associated. A particular instance of an attribute is a value. The domain of an attribute is the collection of all possible values an attribute can have. The domain of Name is a character string.

Attributes can be classified as identifiers or descriptors. Identifiers, more commonly called keys, uniquely identify an instance of an entity. A descriptor describes a non-unique characteristic of an entity instance.

Attribute–property or characteristic of an entity or relationship type (often corresponds to a field in a table)

Relationships:

Relationship instance–link between entities (corresponds to primary key-foreign key equivalencies in related tables)



Notations Used for ER Model:

Entities are represented by labeled rectangles. The label is the name of the entity. Entity names should be singular nouns.
Relationships are represented by a solid line connecting two entities. The name of the relationship is written above the line. Relationship names should be verbs.

Attributes, when included, are listed inside the entity rectangle. Attributes which are identifiers are underlined. Attribute names should be singular nouns.

Cardinality of many is represented by a line ending in a crow's foot. If the crow's foot is omitted, the cardinality is one.

Existence is represented by placing a circle or a perpendicular bar on the line. Mandatory existence is shown by the bar (looks like the number 1) next to the entity that has a mandatory instance. Optional existence is shown by placing a circle next to the entity that is optional.

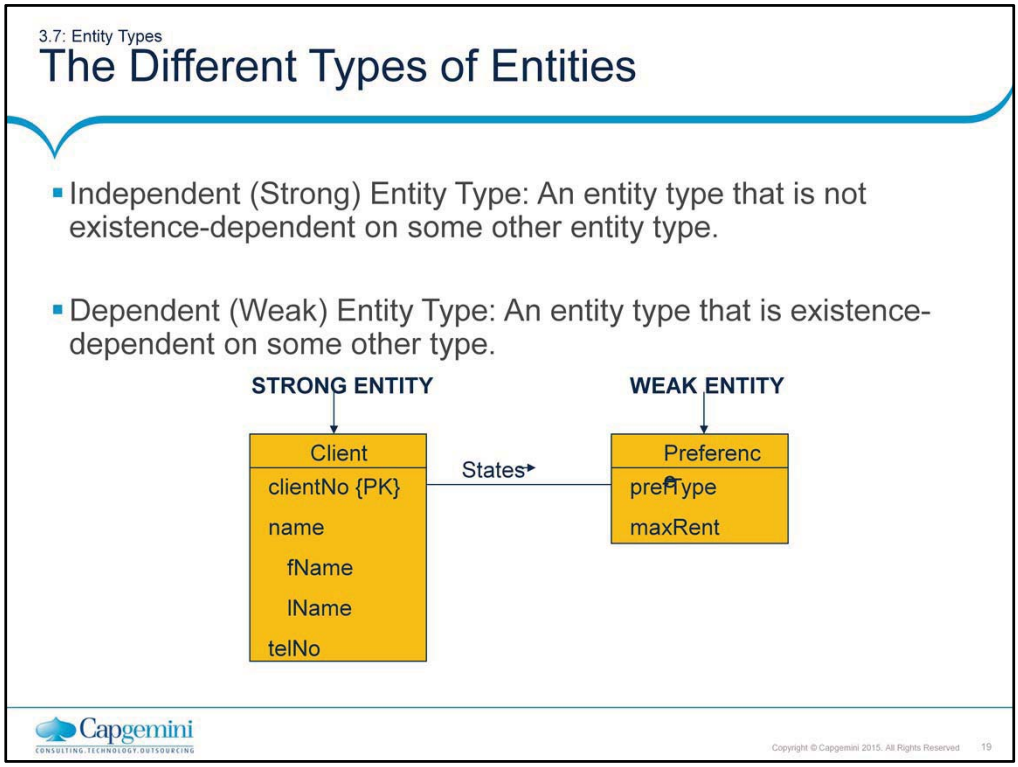
3.7: About Entities

About entities

- An entity is a person, place, thing, event or any of the interest to the enterprise, about which facts may be recorded.
- You should name it in a real world term.
- Eventually entity becomes a table in relational database
- Examples
  - Employee
  - Region
  - Department
  - Customer

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

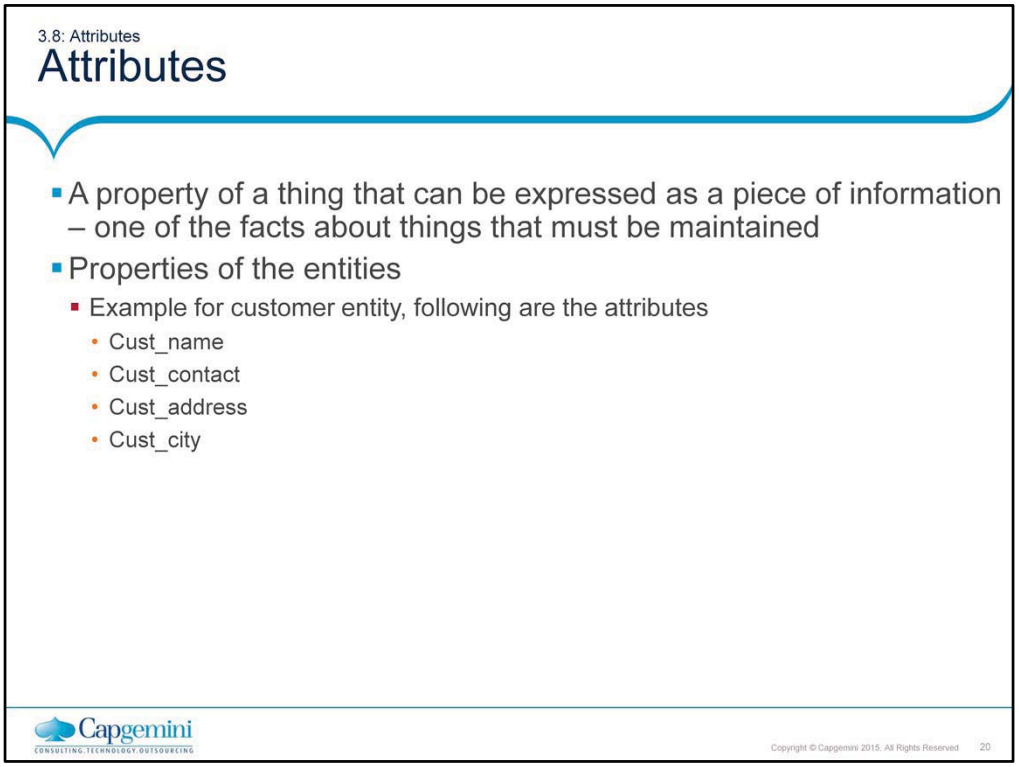
 18



Entity Types

Entities are classified as independent or dependent (in some methodologies, the terms used are strong and weak, respectively). An *independent entity* is the one that does not rely on another for identification. A *dependent entity* is the one that relies on another for identification.

An *entity occurrence* (also called an instance) is an individual occurrence of an entity. An occurrence is analogous to a row in the relational table.



Attributes

Attributes describe the entity with which they are associated. A particular instance of an attribute is a value. For example, “S Ranjan" is one of the values of the attribute Name. The domain of an attribute is the collection of all possible values an attribute can have. The domain of Name is a character string.

Attributes can be classified as identifiers or descriptors. Identifiers, more commonly called keys, uniquely identify an instance of an entity. A descriptor describes a non-unique characteristic of an entity instance.

Types of Attributes

Simple Attributes	
•Each entity has a single atomic value for the attribute.	E.g. SSN or Sex
Composite Attributes	
•The attribute may be composed of several components. •Composition may form a hierarchy where some components are themselves composite	E.g.: Address (Apt#, House#, Street, City, State, Zip_Code, Country) or Name (First_Name, Middle_Name, Last_Name).
Multi-valued Attributes	
•An entity may have multiple values for the attribute.	E.g.: Color of a CAR or Previous Degrees of a STUDENT.
Nested Attributes	
In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels although this is rare.	E.g.: Previous Degrees of a STUDENT is a composite multi-valued attribute denoted by {Previous Degrees (College, Year, Degree, Field)}.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

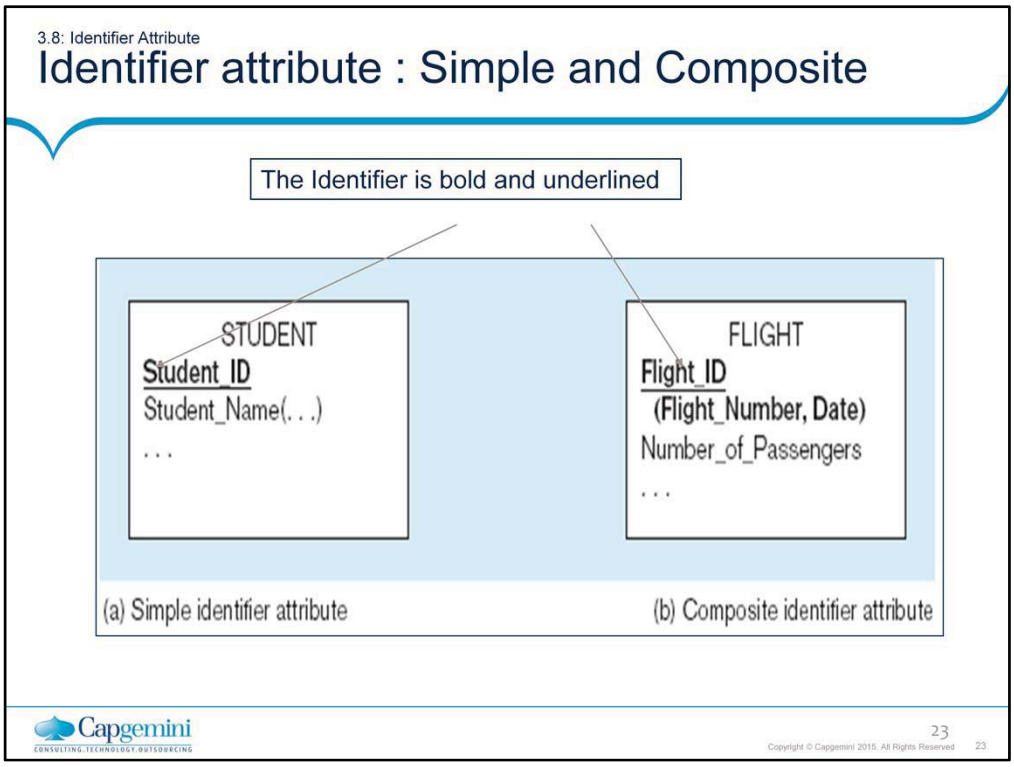
3.8: Identifying Key Attributes

Identifying Key Attributes

- Candidate Key (never NULL): The minimal set of attributes that uniquely identifies each occurrence of an entity type. e.g: branchNo in entity Branch.
- Primary Key: The candidate key that is selected to uniquely identify each occurrence of an entity type. E.g: National Insurance Number.
- Composite Key: A candidate key that consist of two or more attributes.

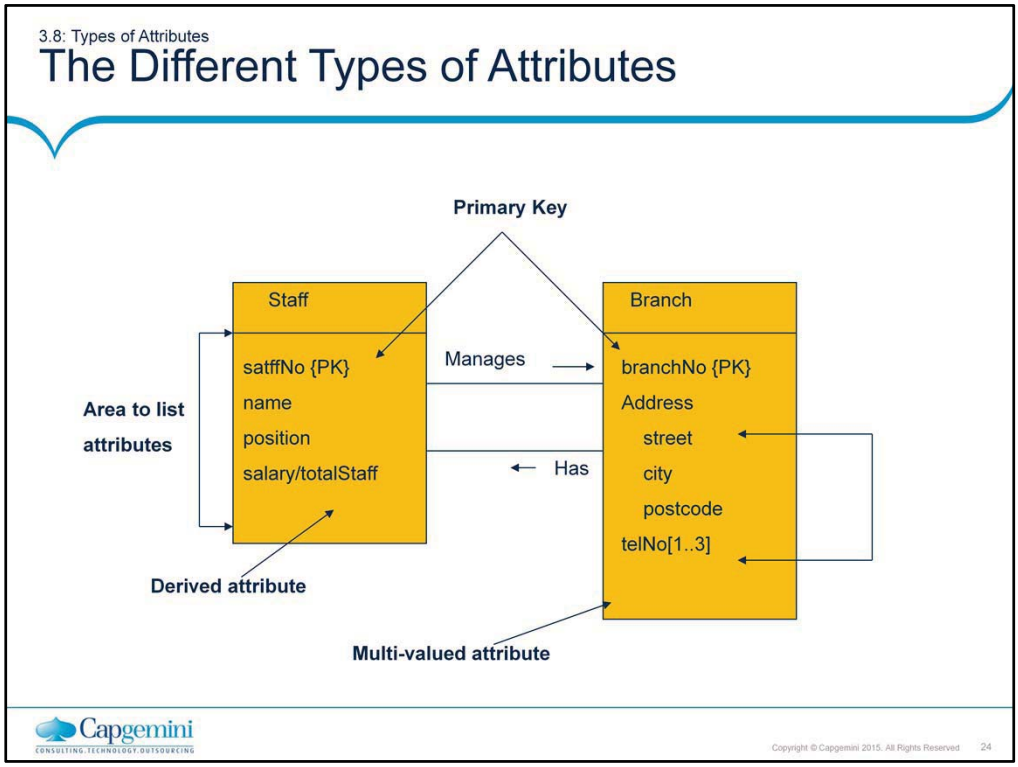
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22



Simple Attribute: An attribute composed of a single component with an independent existence. E.g position and salary of the Staff entity.

Composite Attribute: An attribute composed of multiple components, each with an independent existence. E.g address attribute of the branch entity that can be subdivided into street, city and postcode attributes



Single-Valued Attribute: An attribute that holds a single value for each occurrence. e.g. EmpId

Multi-Valued Attributes: An attribute that holds multiple values for each occurrence. e.g PhoneNo

Derived Attributes: An attribute that represents a value that is derivable from the value of a related attribute or set of attributes, not necessarily in the same entity type. e.g attribute Age whose value is derived from the CurrentDate and DateOfBirth attributes.

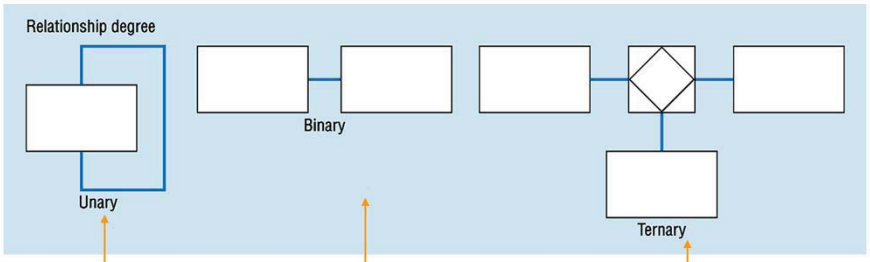
Classifying relationships

- An association between two things (entities) is called a relation.
- Relations are classified by their
  - Degree
  - Connectivity
  - Cardinality
  - Direction
  - Existence.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25

Degree of Relationship



One entity related to another of the same entity type

Entities of two different types related to each other

Entities of three different types related to each other

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26

3.9: Relationships in the RDBMS

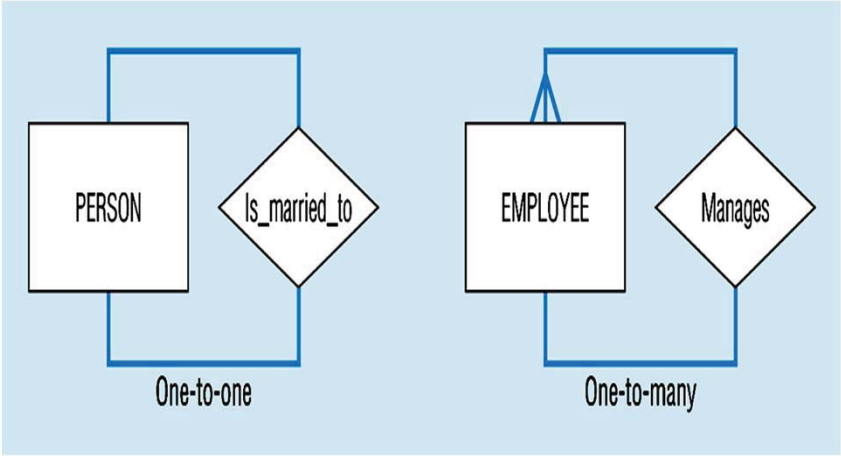
Connectivity and Cardinality

- We have three different types of relationships in RDBMS
  - 1:1 (One to One) – rare
  - 1:M (One to Many) – common
  - M:M (Many to Many) – more in conceptual model, none in Logical model and Physical model
- Examples
  - 1:1 (Person to PAN ID)
  - 1:M (Customer to Phone)
  - M:M (Doctor and Patient)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 27

Cardinality…

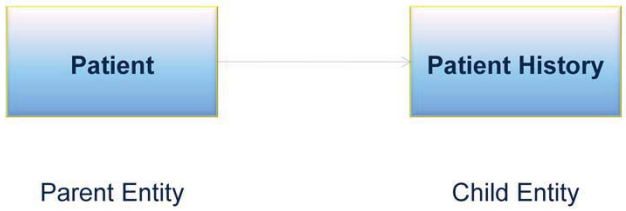


Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 28

direction

- The direction of a relationship indicates the originating entity of a binary relationship.
- The entity from which a relationship originates is the parent entity.
- The entity where the relationship terminates is the child entity.



Parent Entity Child Entity

Existence

- Denotes whether the existence of an entity instance is dependent upon the existence of another, related, entity instance.
- Either mandatory or optional.
- Mandatory - “Every project must be managed by a single department".
- Optional - "employees may be assigned to a BU".

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 30

Constraints on Relationships

- Cardinality Constraints - the number of instances of one entity that can or must be associated with each instance of another entity.
- Minimum Cardinality(also called participation constraint or existence dependency constraints)
  - If zero, then optional participation, not existence-dependent
  - If one or more, then mandatory, existence-dependent
- Maximum Cardinality

The maximum number

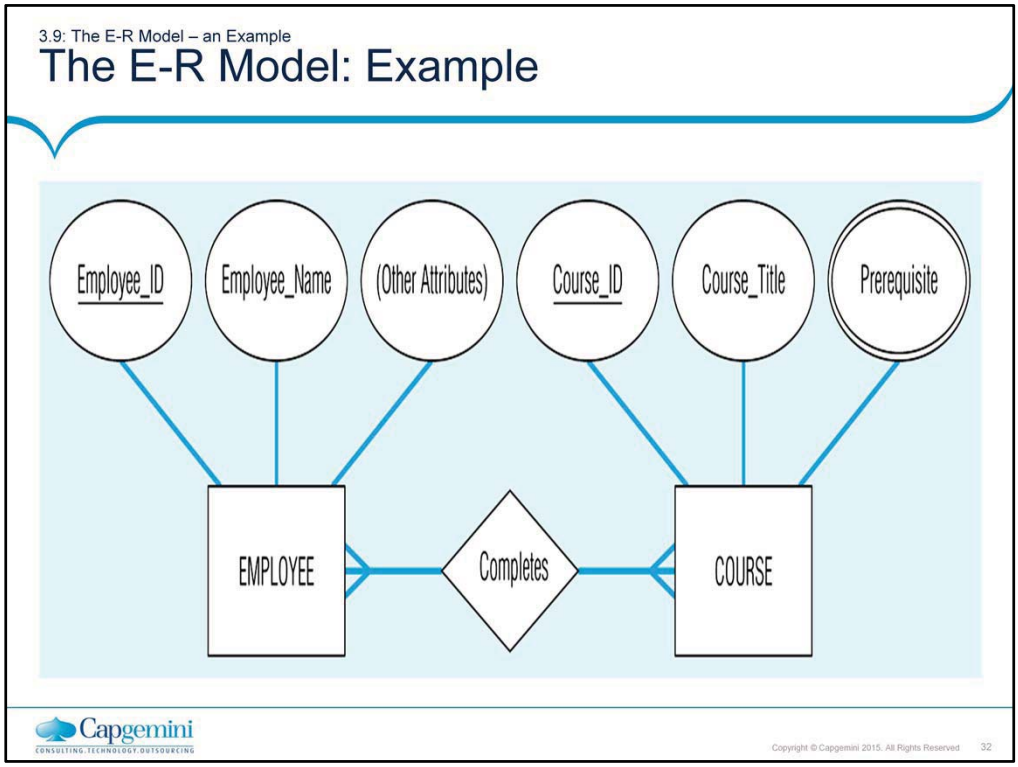
One-to-one (1:1)

One-to-many (1:N) or Many-to-one (N:1)

Many-to-many

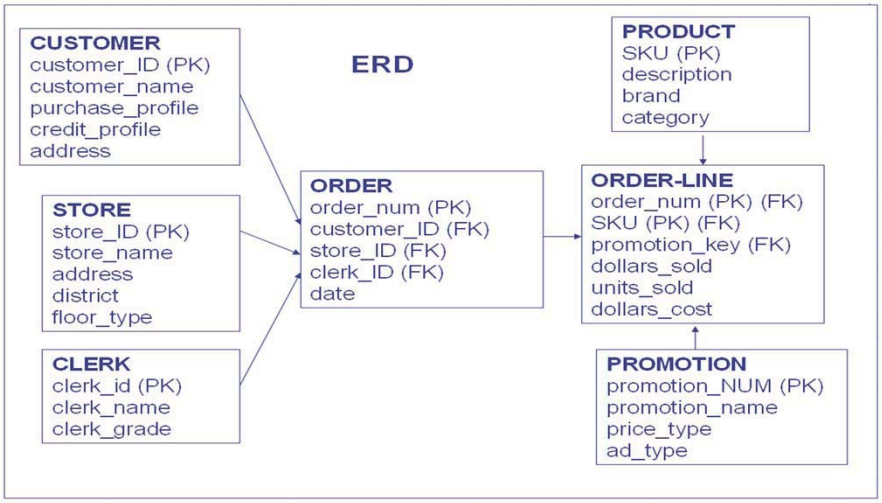
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 31


3.9: The E-R Model – an Example
The E-R Model: Example
Employee_ID
Employee_Name
(Other Attributes)
Course_ID
Course_Title
Prerequisite
EMPLOYEE
Completes
COURSE
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
32

3.9: The E-R Model – an Example

The E-R Model: Example



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 33

Case Study

- The XYZ Company wants Capgemini India Pvt Ltd., to design and develop a database system for its regular operations.
- The database should record information about the departments, projects, employees and their dependant. The company is organized into departments. Employees work for a department and may work on many projects. Departments control the project which are being operated from that location. Department has to be managed by someone.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

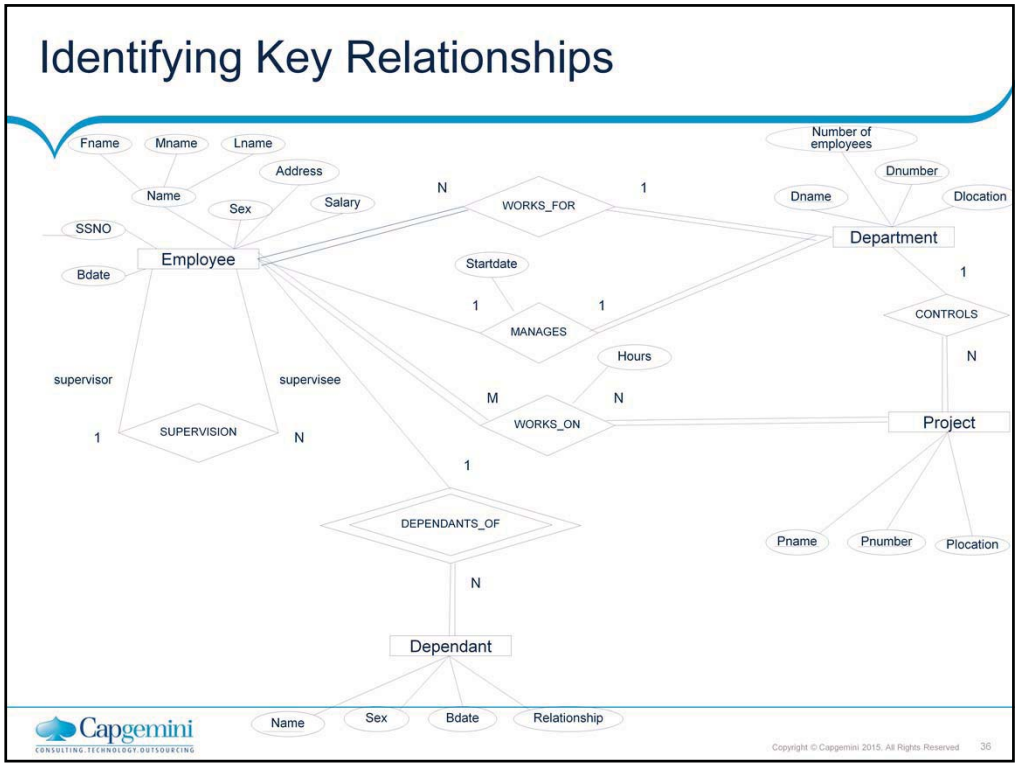
 34

Case Study

- There are managers who manages and monitors the work done by the employees. Suppose an employee is assigned to a project, the hours are calculated based on number of hours the employee is scheduled to work on a project.
- Although most employees have managers, senior staff. The date on which a manager started managing the department could be stored as an attribute of department.
- A department may be spread over many locations. The department name and number are unique for the department. Employee may have number of dependants.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 35


Identifying Key Relationships
Fname
Mname
Lname
Address
Name
Sex
Salary
SSNO
Employee
Bdate
N
WORKS_FOR
1
Number of employees
Dnumber
Dname
Dlocation
Department
Startdate
1
1
MANAGES
1
CONTROLS
N
Hours
supervisor
supervisee
M
N
WORKS_ON
Project
1
SUPERVISION
N
1
DEPENDANTS_OF
Pname
Pnumber
Plocation
N
Dependant
Name
Sex
Bdate
Relationship
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
36

1-to-1 Entity Relationship

- The relationship between these two entities is 1 to 1 because in this company, only 1 manager is allowed to manage a single department.
- Every department is required to have an assigned manager.
- What kind of table design does this suggest?
- A single table: for the Department entity that includes the Manager Entity.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 37

One-to-many (1:N) RELATIONSHIP

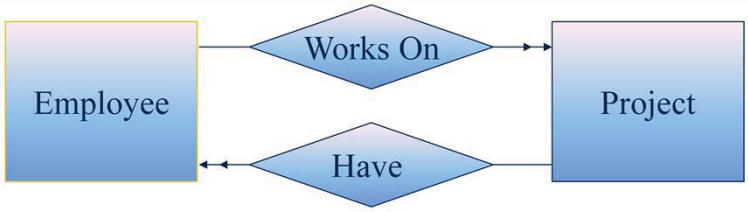
- The relationship between these two entities is 1 to Many because there can be 1 or more employees in each department.
- Every department is required to have at least one employee, and no employee can belong to more than one department.
- What kind of table design does this suggest?
- A single table for each entity: the Department Table and Employee Table.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Many-to-one (N:1) RELATIONSHIP

- The relationship between these two entities is Many to 1 because there can be 1 or more dependants for each employee.
- What kind of table design does this suggest?
- A single table for each entity: the Dependants Table and Employee Table.

Many- to – Many (N:M) Relationship

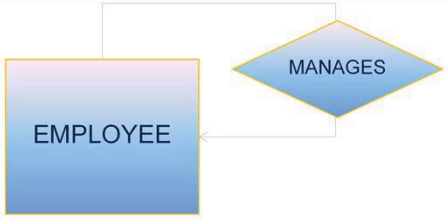


- These 2 entities have 2 relationships - 1 to many in each direction - resulting in a many-many relationship.
- Employees are optionally assigned to one or more Projects, as appropriate. A Project must have at least 1 employee.
- What kind of table design does this suggest?
- 2 Tables plus a table with a column for each entity. (Employee, Project, Employee_Project)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 40

Recursive Relationships

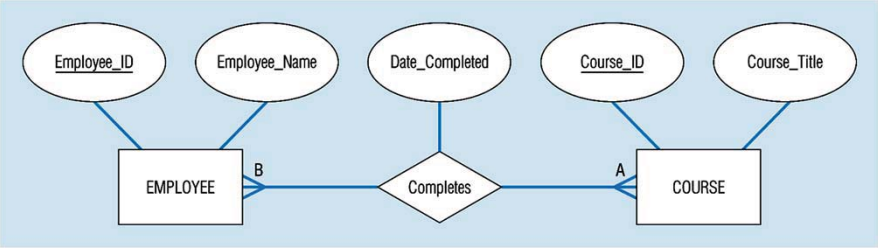


- We can also have a recursive relationship type.
- Both participations are same entity type in different roles.
  E.g.: SUPERVISION (MANAGES) relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker).
- In ER diagram, need to display role names to distinguish Participations.

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

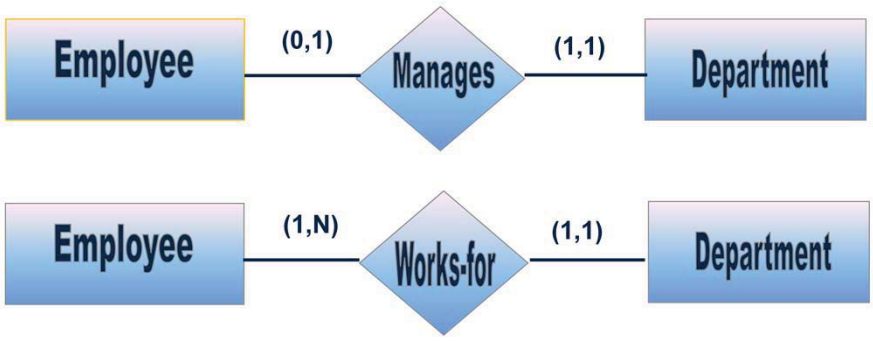
 41

Attributes of Relationship types



- Here, the date completed attribute pertains specifically to the employee's completion of a course…it is an attribute of the relationship

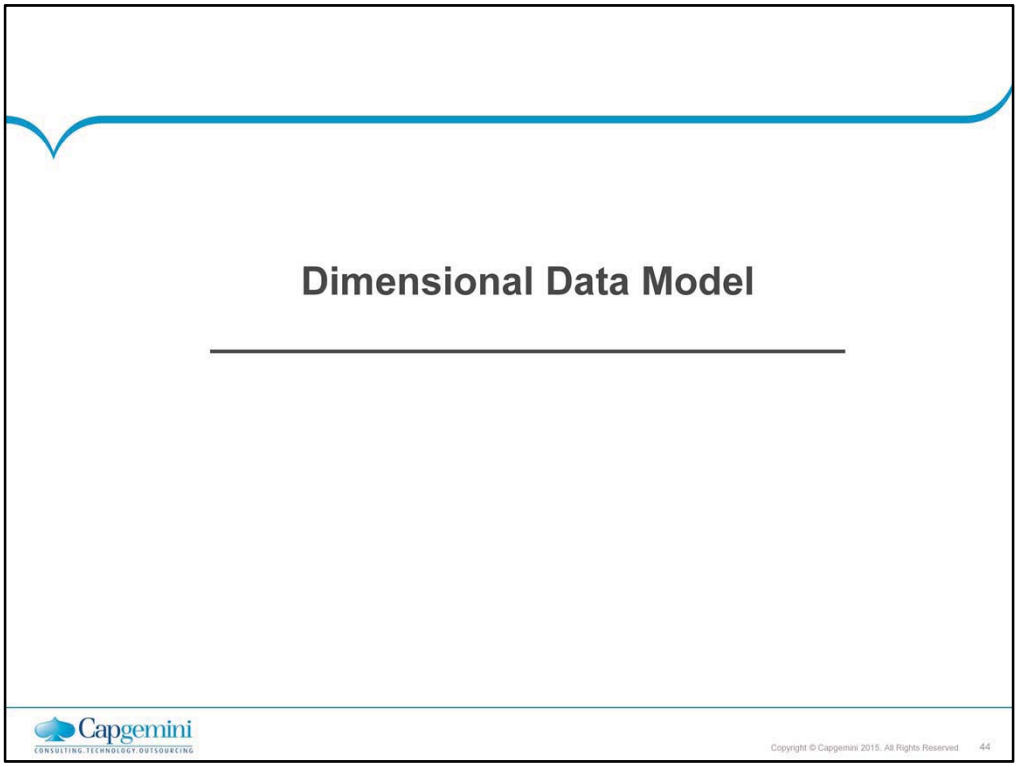
Notation



- The (min, max) notation relationship constraints

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 43


Dimensional Data Model
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved. 44

3.10: Dimension Modeling

What is dimension modeling?

- Dimension modeling is used to model for data warehouses and data marts, and is different from OLTP modeling.
- Data mart/warehouse differs from OLTP on the basis of the following:
  - Usage (It is information-driven rather than transaction-driven)
  - Type of database used (Multi-dimensional rather than relational)

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 45

Dimensional Model

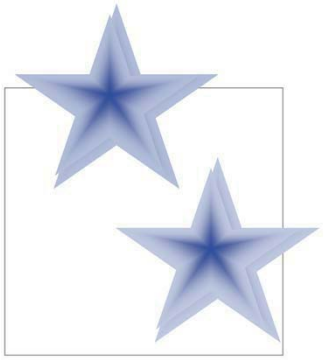
- Definition
  - Logical data model used to represent the measures and dimensions that pertain to one or more business subject areas
  - Dimensional Model = Star Schema
- Serves as basis for the design of a relational database schema
- Can easily translate into multi-dimensional database design if required
- Overcomes OLTP design shortcomings

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 46

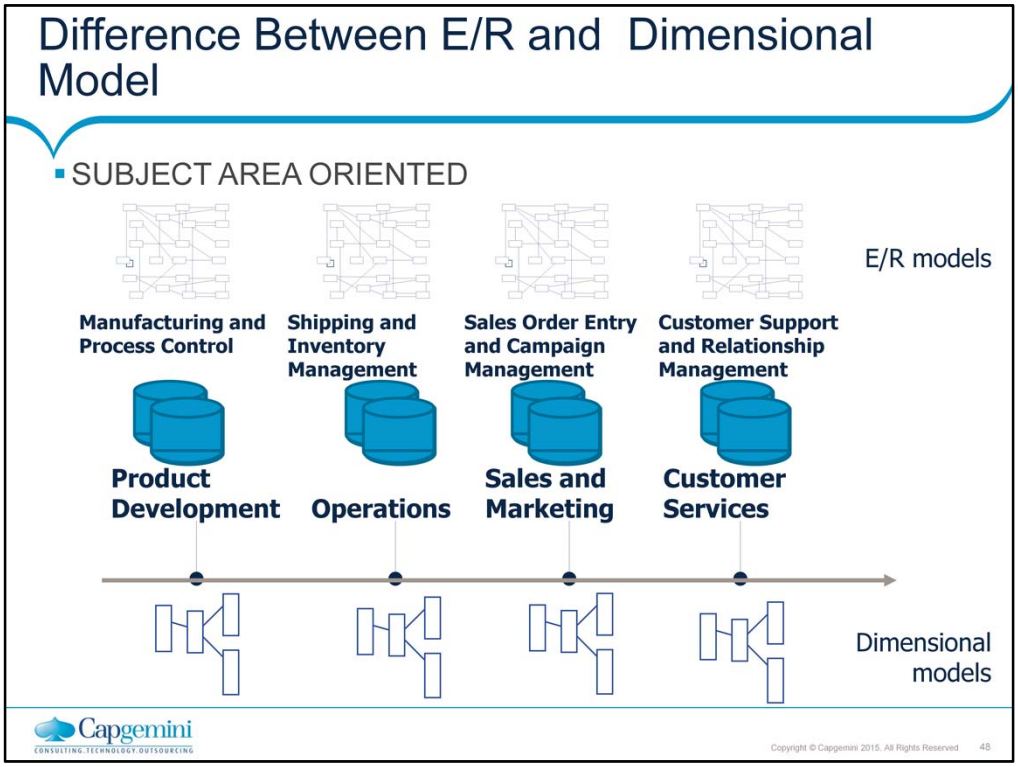
Dimensional Model Advantages

- Understandable
- Systematically represents history
- Reliable join paths
- High performance query
- Enterprise scalability



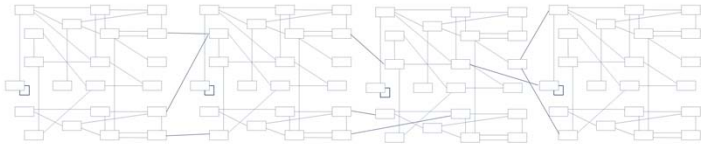
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 47

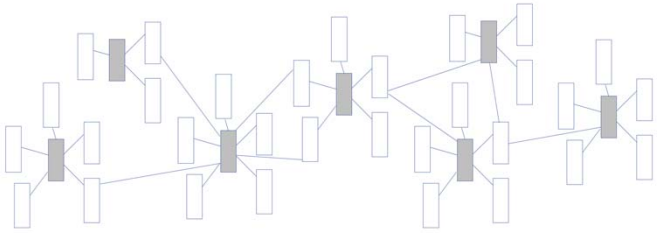

Difference Between E/R and Dimensional Model
SUBJECT AREA ORIENTED
E/R models
Manufacturing and Process Control
Shipping and Inventory Management
Sales Order Entry and Campaign Management
Customer Support and Relationship Management
Product Development
Operations
Sales and Marketing
Customer Services
Dimensional models
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
48

Difference Between E/R and Dimensional Model

- ENTERPRISE SCOPE FOR MODELS



E/R model



Dimensional model

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 49

Process Measurement

- Measures
  - Metrics or indicators by which people evaluate a business process
  - Referred to as “Facts”
- Examples
  - Margin
  - Inventory Amount
  - Sales Dollars
  - Receivable Dollars
  - Return Rate

Coffee Maker Fulfillment Report

Brand	Product	Units Sold	Units Shipped	% Shipped
Captain Coffee	Standard Coffee Maker	5,000	3,800	76%
	Thermal Coffee Maker	2,400	1,632	68%
	Deluxe Coffee Maker	2,073	1,658	80%
	All Products	9,473	**7,090**	75%

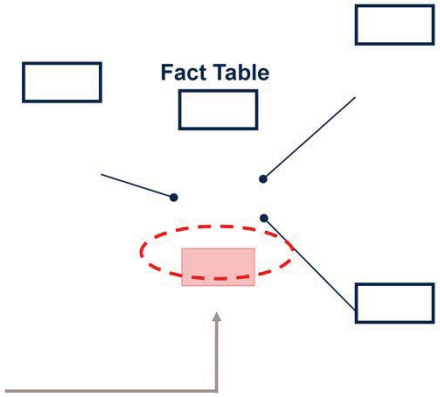
Facts

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 50

Fact Table Grain

- Grain
  - The level of detail represented by a row in the fact table
  - Must be identified early
  - Cause of greatest confusion during design process
- Example
  - Each row in the fact table represents the daily item sales total



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 51

Process Perspectives

- Dimensions
  - The parameters by which measures are viewed
  - Used to break out, filter or roll up measures
  - Often found after the word “by” in a business question
  - Descriptive business terms
- Examples
  - Product
  - Warehouse
  - Customer
  - Supplier

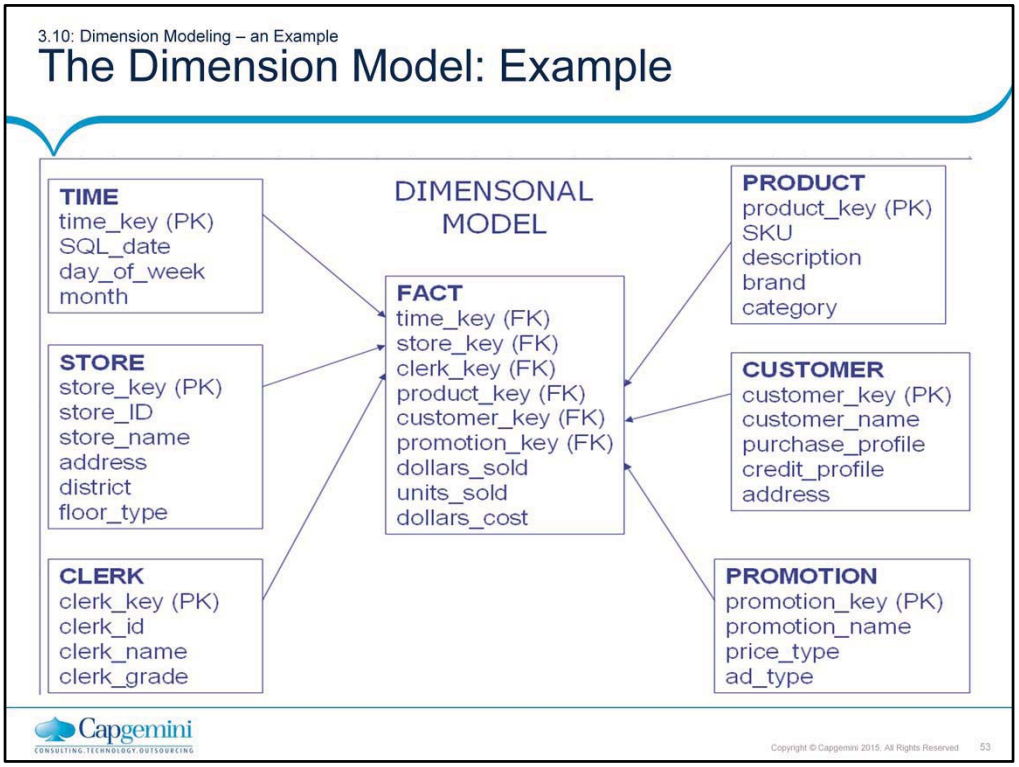
Coffee Maker Fulfillment Report

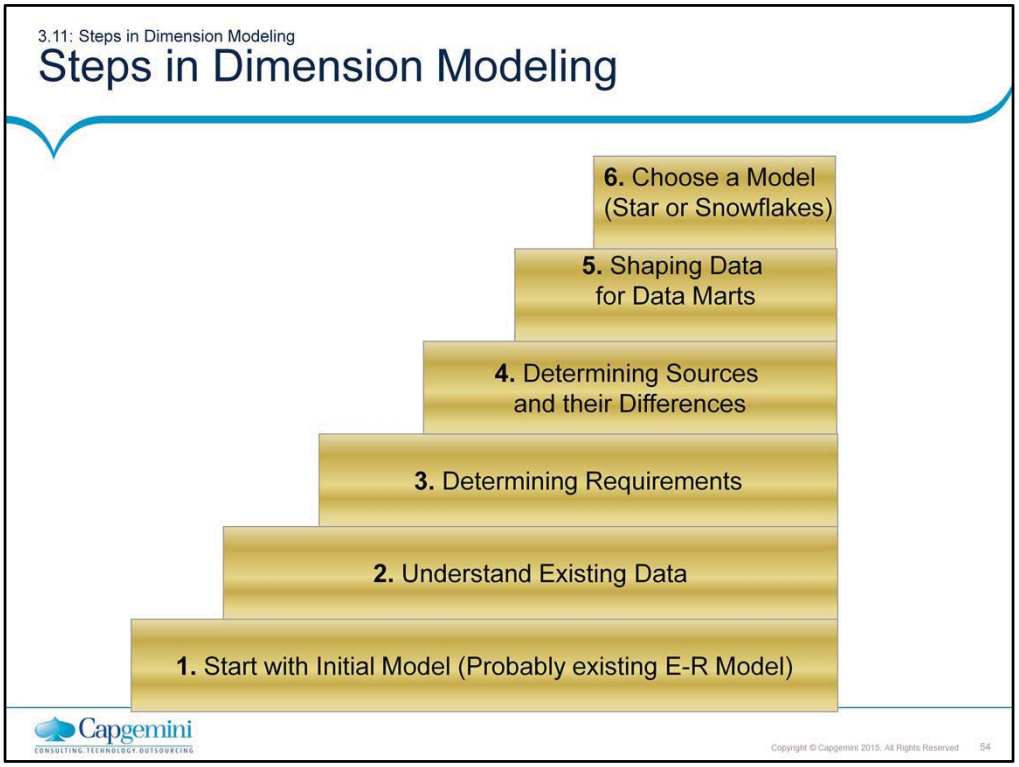
Brand	Product	Units Sold	Units Shipped	% Shipped
Captain Coffee	Standard Coffee Maker	5,000	3,800	76%
	Thermal Coffee Maker	2,400	1,632	68%
	Deluxe Coffee Maker	2,073	1,658	80%
	All Products	**9,473**	**7,090**	**75%**

Dimensions

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

 52

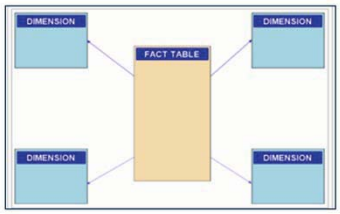

3.10: Dimension Modeling – an Example
The Dimension Model: Example
DIMENSONAL MODEL
TIME
time_key (PK)
SQL_date
day_of_week
month
STORE
store_key (PK)
store_ID
store_name
address
district
floor_type
CLERK
clerk_key (PK)
clerk_id
clerk_name
clerk_grade
FACT
time_key (FK)
store_key (FK)
clerk_key (FK)
product_key (FK)
customer_key (FK)
promotion_key (FK)
dollars_sold
units_sold
dollars_cost
PRODUCT
product_key (PK)
SKU
description
brand
category
CUSTOMER
customer_key (PK)
customer_name
purchase_profile
credit_profile
address
PROMOTION
promotion_key (PK)
promotion_name
price_type
ad_type
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
53


3.11: Steps in Dimension Modeling
Steps in Dimension Modeling
6. Choose a Model (Star or Snowflakes)
5. Shaping Data for Data Marts
4. Determining Sources and their Differences
3. Determining Requirements
2. Understand Existing Data
1. Start with Initial Model (Probably existing E-R Model)
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
54

3.12: The Star Schema

The Star Schema

- A fact table is surrounded by a set of dimension tables.
- The fact table holds transaction data.
- The dimensions are business objects or entities of interest.
- It's easy to understand.



Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 55

In addition to the measurements, a fact table contains foreign keys for the dimension tables. These foreign keys are connected to the primary keys of the dimension table. . In star schema each dimension is represented by a single table.

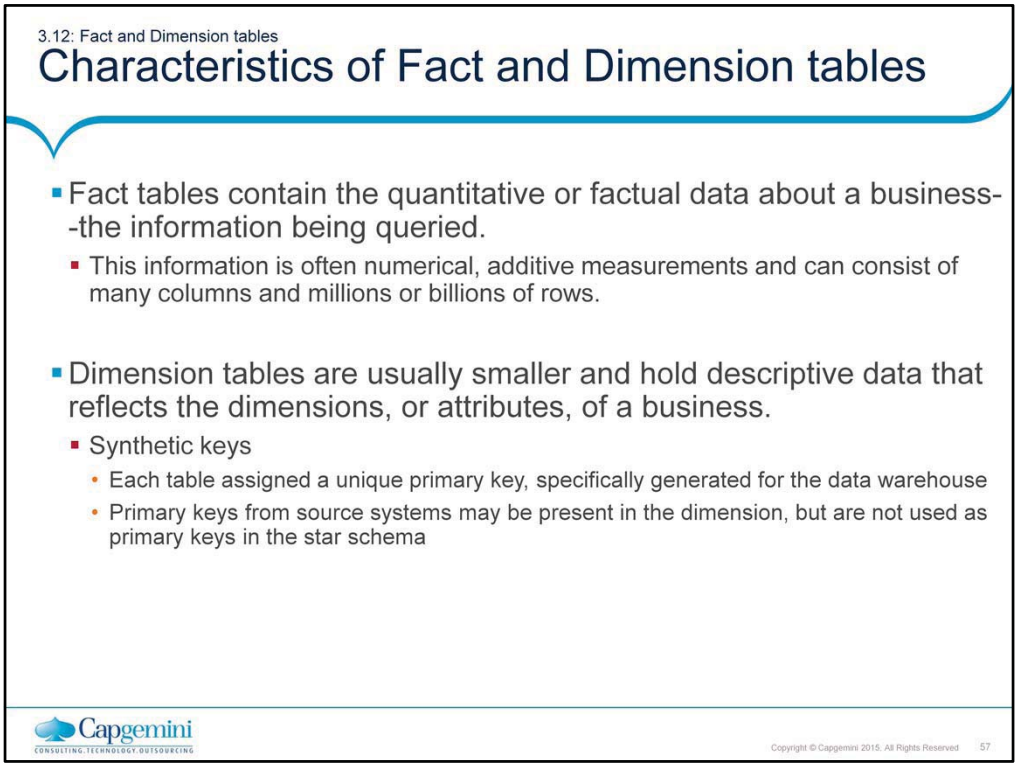
Star Schema is a relational database schema for representing multidimensional data. It is the simplest form of data warehouse schema that contains one or more dimensions and fact tables. It is called a star schema because the entity-relationship diagram between dimensions and fact tables resembles a star where one fact table is connected to multiple dimensions. The center of the star schema consists of a large fact table and it points towards the dimension tables.

Designing a Star Schema

- Based on Kimball's six steps
  - Start designing in order
  - Re-visit and adjust over project life
  - Five initial design steps
    - Identify fact table
    - Identify fact table grain
    - Identify dimensions
    - Select facts
    - Identify dimensional attributes

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 56



Fact table characteristics

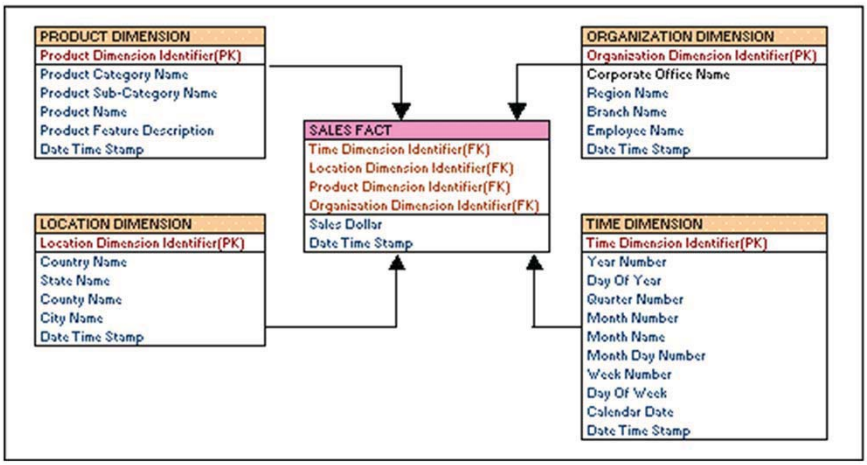
- The fact table contains numerical values of what you measure. For example, fact value of 20 might mean that 20 widgets have been sold.
- Each fact table contains the keys to associated dimension tables. These are called *foreign keys* in the fact table.
- Fact tables typically contain a small number of columns.
- Compared to dimension tables, fact tables have a large number of rows.
- The information in a fact table has characteristics, such as:
  - It is numerical and used to generate aggregates and summaries.
  - Data values need to be additive, or semi-additive, to enable summarization of a large number of values.

Dimension table characteristics

- Dimension tables contain the details about the facts. That, as an example, enables the business analysts to better understand the data and their reports.
- The dimension tables contain descriptive information about the numerical values in the fact table. That is, they contain the attributes of the facts. For example, the dimension tables for a marketing analysis application might include attributes such as time period, marketing region, and product type.
- Since the data in a dimension table is denormalized, it typically has a large number of columns.
- The dimension tables typically contain significantly fewer rows of data than the fact table.
- The attributes in a dimension table are typically used as row and column headings in a report or query results display. For example, the textual
- descriptions on a report come from dimension attributes.

3.12: Star Schema – an Example

Star Schema: Example



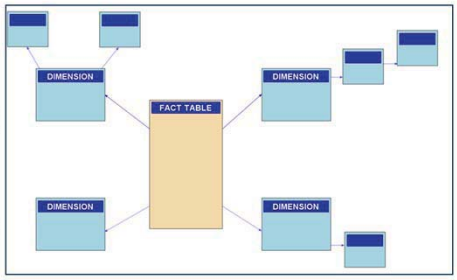
Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 58

3.12: The Snow Flake Schema

The Snow Flake Schema

- A fact table is surrounded by a set of Normalized Dimension tables.
- Dimension tables are broken into hierarchical tables.
- This increases number of Joins.
- The table may become large and unmanageable.

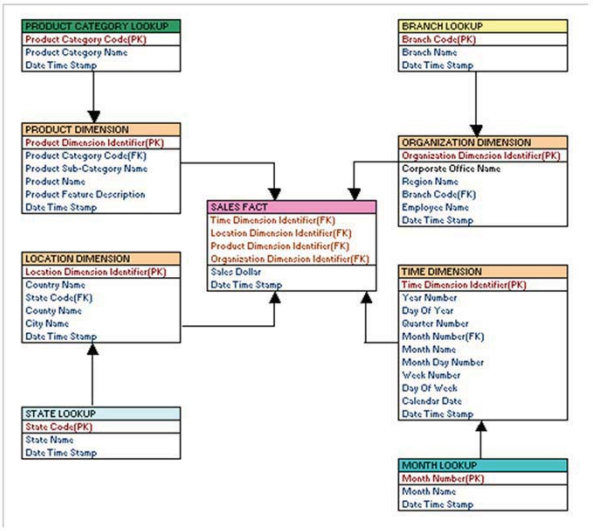


Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 59

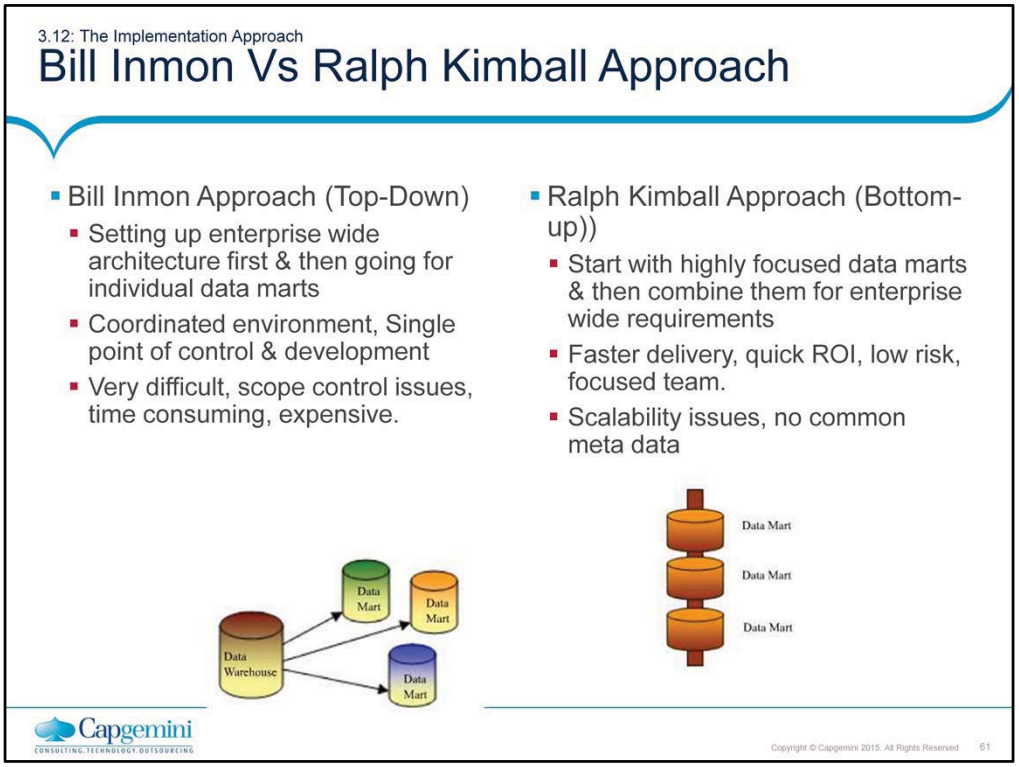
3.12: Snow Flake Schema - Example

Snow Flake Schema: Example



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 60



Both experts are of the opinion that the success of the warehouse/marts depends on effectively gathering the business requirements first. These requirements drive the design of the mart which, in turn, drives the data required in the warehouse. Both experts agree that business-user validation of the data mart design ensures that expectations are managed.

The initial model is the starting point for the design of the staging area (or warehouse). This is where the referential integrity rules are applied (via the DBMS or software validation) and transformation of disparate values is performed. Kimball calls it the backroom, and Inmon calls it the data warehouse.

Inmon advocated a “dependent data mart structure” whereas Kimball advocated the “data warehouse bus structure”.

Bill Inmon Approach
Transfer of data happens from diverse OLTP systems into a centralized place where the data could be used for analysis. Warehouse needs to be build first and data should be made accessible at detailed atomic levels by drilling down or at summarized levels by drilling up. The data marts are treated as sub sets of the data warehouse. Each data mart is built for an individual department and is optimized for their analysis needs.

This data is loaded into the staging area and validated and consolidated for ensuring a level of accuracy and then transferred to the **optional** Operational Data Store (ODS). Data is also loaded into the Data warehouse in a parallel process to avoid extracting it from the ODS. Once the Data warehouse building processes are complete, the data mart refresh cycles will extract the data from the Data warehouse into the staging area and perform a new set of transformations on them. This helps in organizing the data in particular structures required by data marts.

Ralph Kimball Approach
Ralph Kimball suggested the data warehouse with the data marts connected to it with a bus structure. The bus structure contained all the common elements that are used by data marts such as conformed dimensions, measures etc defined for the enterprise as a whole. According to him, by using these conformed elements, users can query all data marts together. This architecture makes the data warehouse more of a virtual reality than a physical reality.
The bottom-up approach reverses the positions of the Data warehouse and the Data marts. Data marts are directly loaded with the data from the operational systems through the staging area.

Hybrid Approach
Start with data mart having focus on enterprise wide scope. It aims to harness the speed and user orientation of the Bottom up approach to the integration of the top-down approach. The Hybrid approach begins with an Entity Relationship diagram of the data marts and a gradual extension of the data marts to extend the enterprise model in a consistent, linear fashion. The data from the various data marts are then transferred to the data warehouse and query tools are reprogrammed to request summary data from the marts and atomic data from the data warehouse.

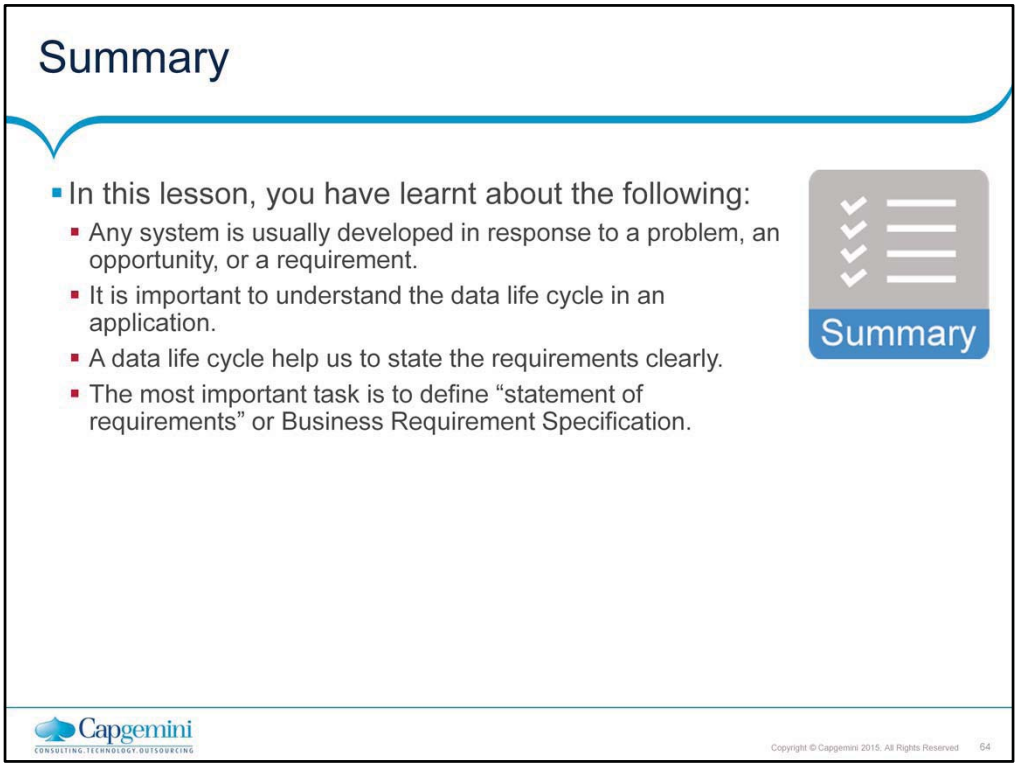
3.13: Conceptual Data Design

Conceptual Data Design

Feature	Conceptual	Logical	Physical
Entity Names	✓		
Entity Relationships	✓		
Attributes			
Primary Keys			
Foreign Keys			
Table Names			
Column Names			
Column Data Types			

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 63



Add the notes here.

Review Question

- Question 1: In ____ a fact table is surrounded by a set of Normalized Dimension tables.
- Question 2: An association between two things (entities) is called a ____.

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 65

Data Modeling for Business Intelligence

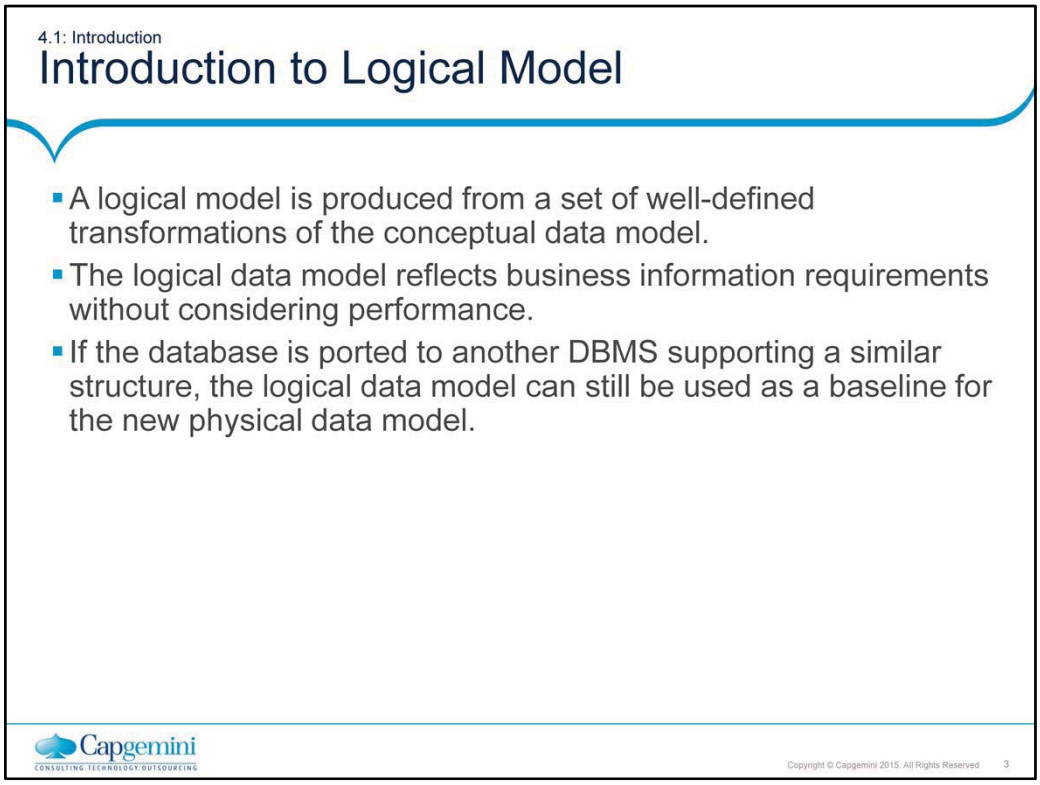
Lesson 4: Logical Model

Lesson Objectives

- On completion of this lesson, you will be able to:
  - Define logical model
  - List features of a logical model
  - Name the transformations required to be done while converting a conceptual model into a logical model
  - Identify activities involved in those transformations
  - Name the types of attributes which do not get converted into a single column in the logical model
  - Data modeling tools

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



The primary purpose of logical data modeling is to document the business information structures, processes, rules, and relationships by a single view.

The logical data model helps to address the following:

1) Validation of the functional application model against business requirements
2) The product and implementation independent requirements for the physical database design (Physical Data Modeling)
3) Clear and unique identification of all business entities in the system along with their relations.

Note:

Without the logical data model, the stored business information is described by a functional model or conceptual model. Without the logical data model, there is no single view of all data, and data normalization is impossible.

In this case, the physical data model has to be designed from a functional model. This will potentially cause performance problems, and data inconsistency and redundancies, which can result in an inefficient physical design.

4.2: Features of a Logical Model

Characteristics of a Logical Model

- Logical model works in an iterative manner.
- Its design is independent of database.
- It includes all entities and relationships among them.
- All attributes for each entity are specified.
- The primary key for each entity is specified.
- Foreign keys (keys identifying the relationship between different entities) are specified.

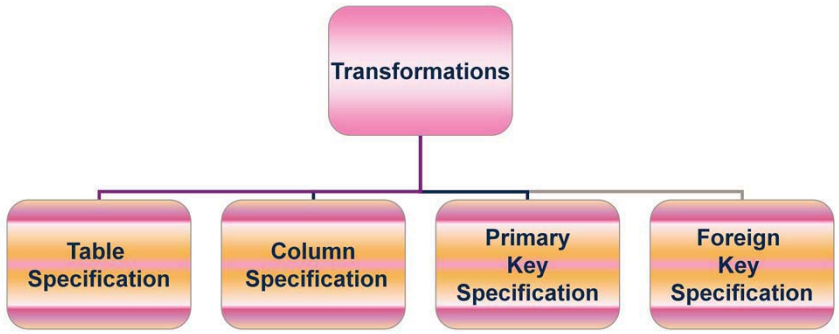
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4

4.3: Requisite Transformations

Transformation Required

- While converting a conceptual model into a logical model, following transformations are required to be performed:



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

4.3: Table Specification

Activities in Table Specification

- In general, each entity class in the conceptual data model becomes a table in the logical data model and is given a name that corresponds to that of the source entity class

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

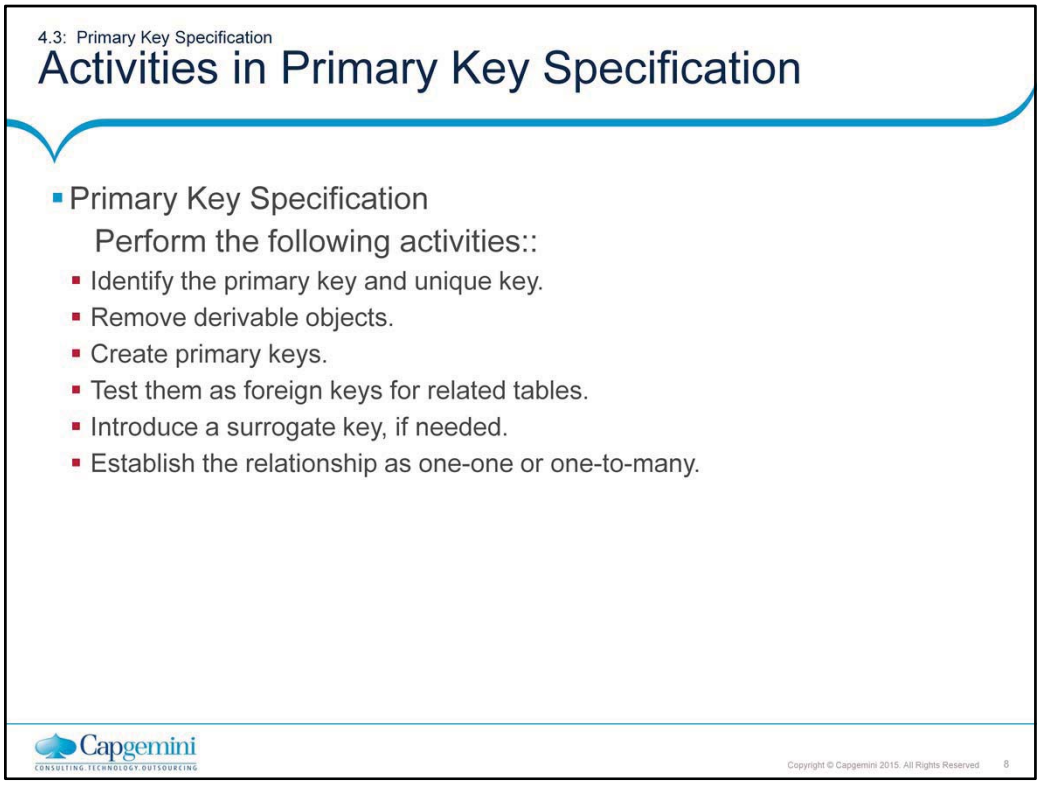
4.3: Column Specification

Activities in Column Specification

- In general, each attribute in the conceptual data model becomes a column in the logical data model and should be given a name that corresponds to that of the corresponding attribute.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7



Activities in Primary Key Specification:

Existing columns are assessed for the primary keys; and if required, surrogate keys are introduced.

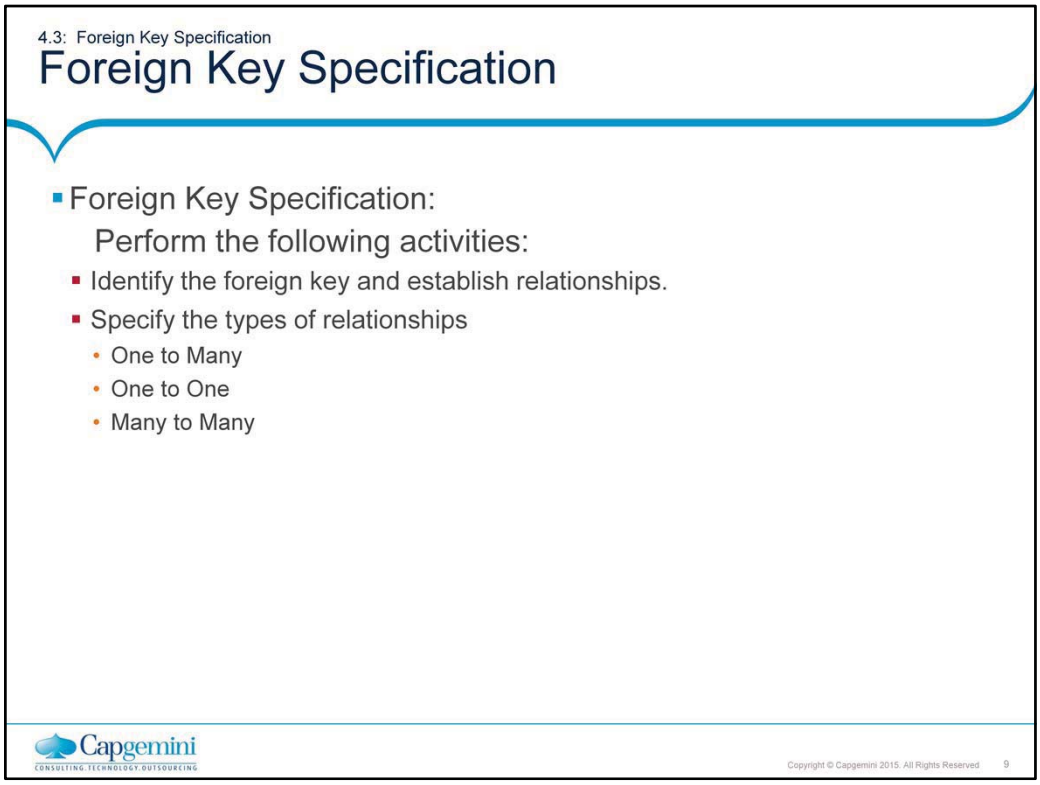
To access data in a relational database, we need to be able to locate specific rows of a table by specifying values for their primary key column or columns.

In particular:

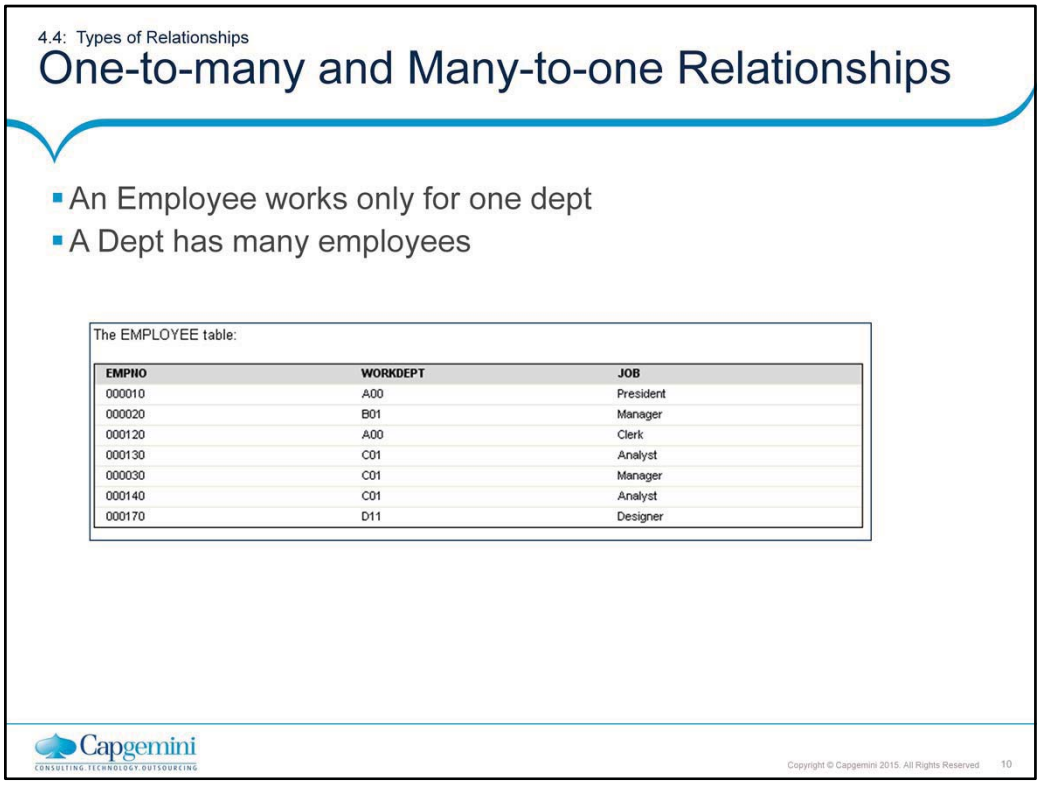
- We must be able to unambiguously specify the row that corresponds to a particular real-world entity instance. When a payment for an account arrives, we need to be able to retrieve the single relevant row in the **Account** table by specifying the Account Number that was supplied with the payment.
- Relationships are implemented using foreign keys, which must each point to one row only. Imagine the problems if we had an insurance policy that referred to customer number “12345” but found two or more rows with that value in the **Customer** table.

So we require that a primary key be *unique*.

- A very simple way of meeting all of the requirements is to invent a new column for each table, specifically to serve as its primary key, and to assign a different system-generated value to each row, and, by extension, to the corresponding entity instance. We refer to such a column as a **surrogate key**, which is typically named by appending “ID” (or, less often, “Number” or “No”) to the table name. Familiar examples are customer IDs, employee IDs, and account numbers allocated by the system.



Foreign keys are our means of implementing one-to-many and occasionally one-to-one relationships.



An employee can work in only one department; this relationship is single-valued for employees. On the other hand, one department can have many employees; this relationship is multi-valued for departments. The relationship between employees (single-valued) and departments (multi-valued) is a one-to-many relationship.

To define tables for each one-to-many and each many-to-one relationship:

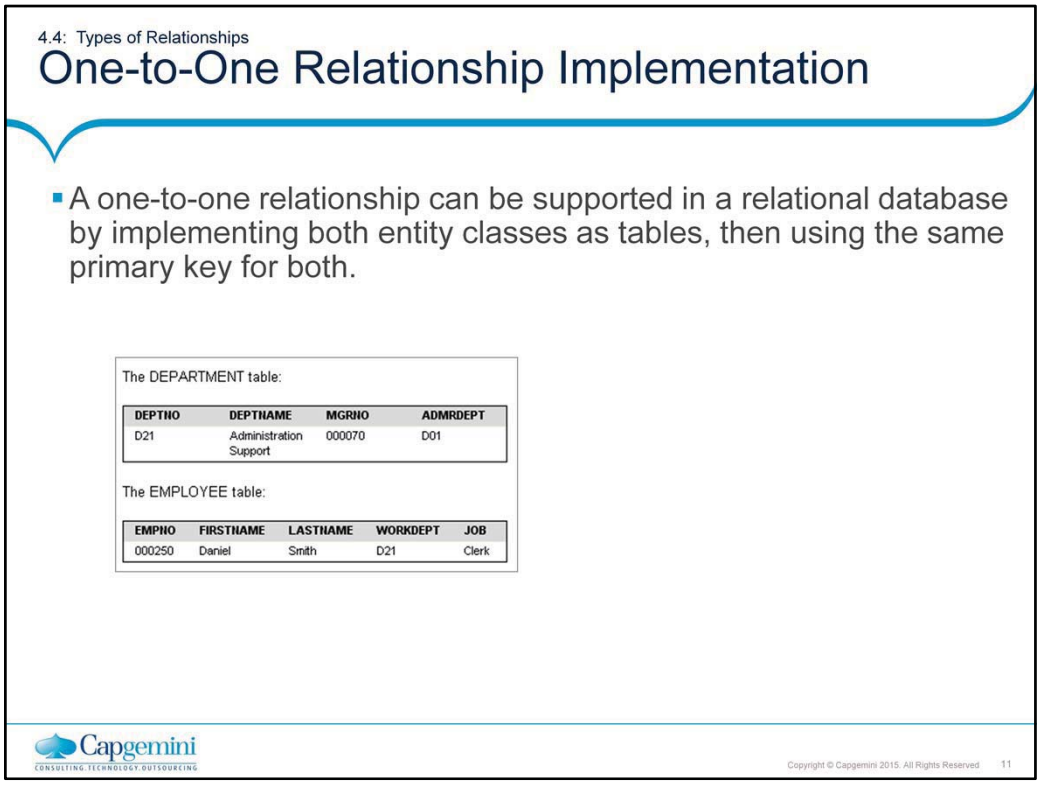
- Group all the relationships for which the "many" side of the relationship is the same entity.
- Define a single table for all the relationships in the group.

One-to-Many Relationship Implementation

When moving from a conceptual to a logical data model, however, we work from a diagram to tables and apply the following rule

A one-to-many relationship is supported in a relational database by holding the primary key of the table representing the entity class at the "one" end of the relationship as a foreign key in the table representing the entity class at the "many" end of the relationship.

In the logical data model, therefore, we create, in the table representing the entity class at the "many" end of the relationship, a copy of the primary key of the entity class at the "one" end of the relationship.



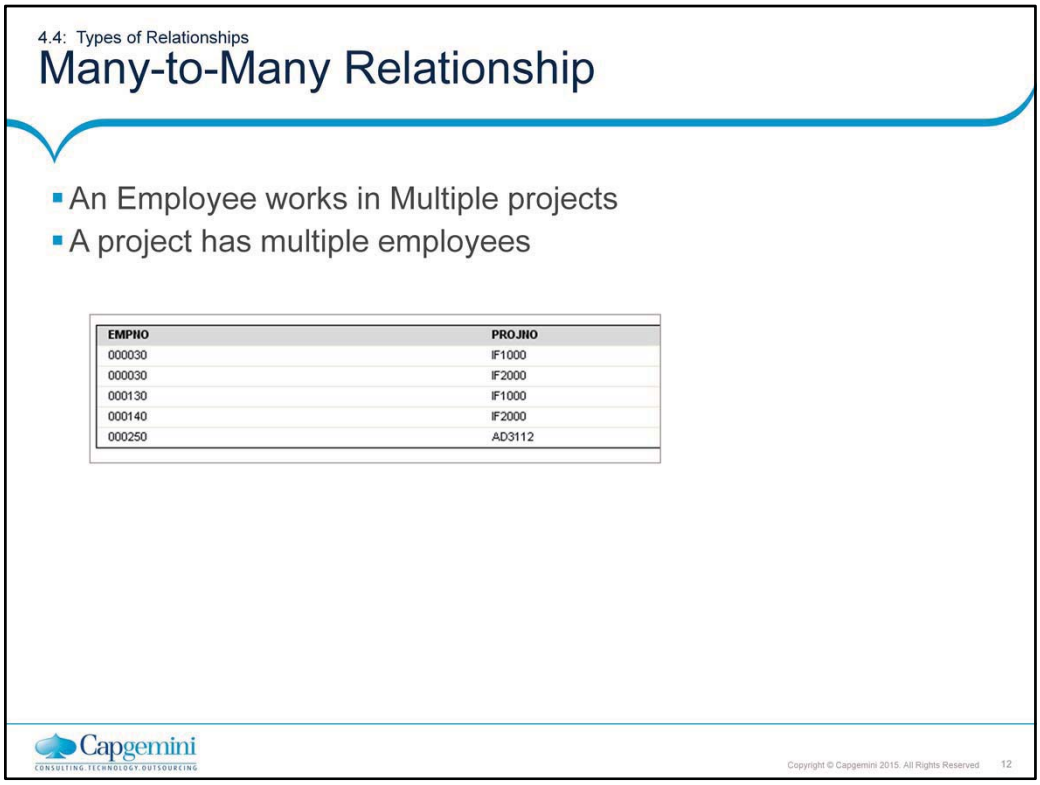
One-to-one relationships are single-valued in both directions. A manager manages one department; a department has only one manager. The questions, "Who is the manager of Department C01?", and "What department does Sally Kwan manage?" both have single answers.

The relationship can be assigned to either the DEPARTMENT table or the EMPLOYEE table. Because all departments have managers, but not all employees are managers, it is most logical to add the manager to the DEPARTMENT table, as shown in the following example.

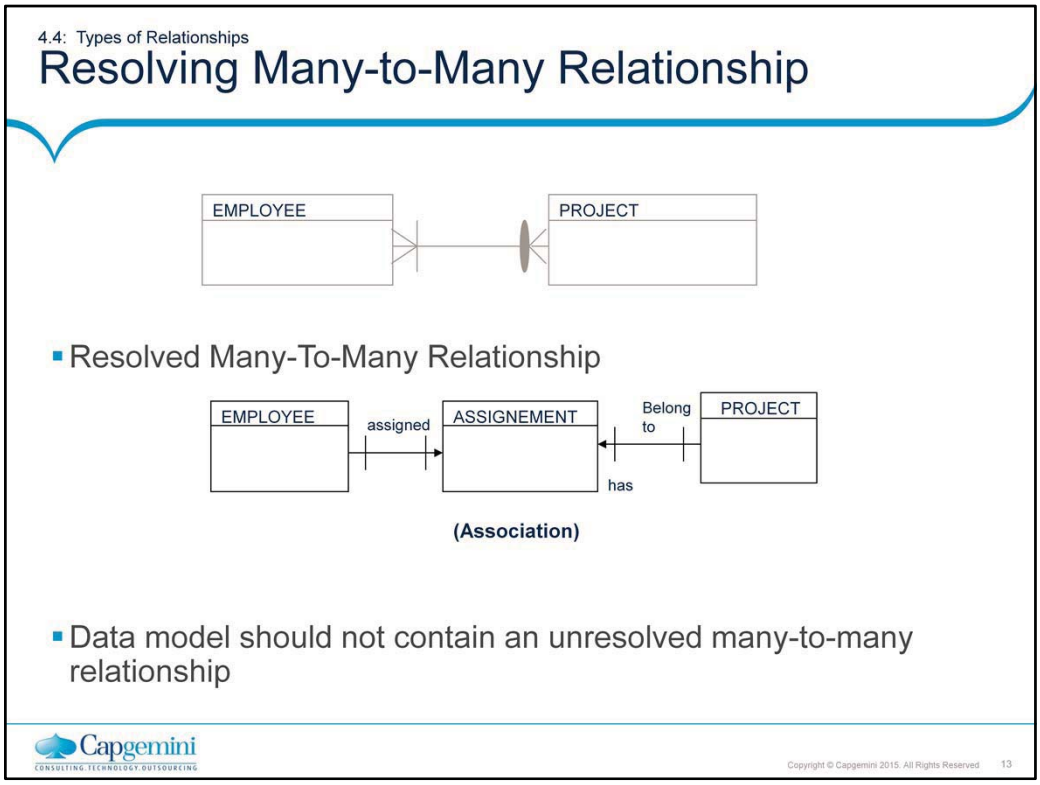
The above table shows the representation of a one-to-one relationship.

You can have more than one table describing the attributes of the same set of entities. For example, the EMPLOYEE table shows the number of the department to which an employee is assigned, and the DEPARTMENT table shows which manager is assigned to each department number. To retrieve both sets of attributes simultaneously, you can join the two tables on the matching columns, as shown in the following example. The values in WORKDEPT and DEPTNO represent the same entity, and represent a *join path* between the DEPARTMENT and EMPLOYEE tables.

When you retrieve information about an entity from more than one table, ensure that equal values represent the same entity. The connecting columns can have different names (like WORKDEPT and DEPTNO in the previous example), or they can have the same name (like the columns called DEPTNO in the department and project tables).



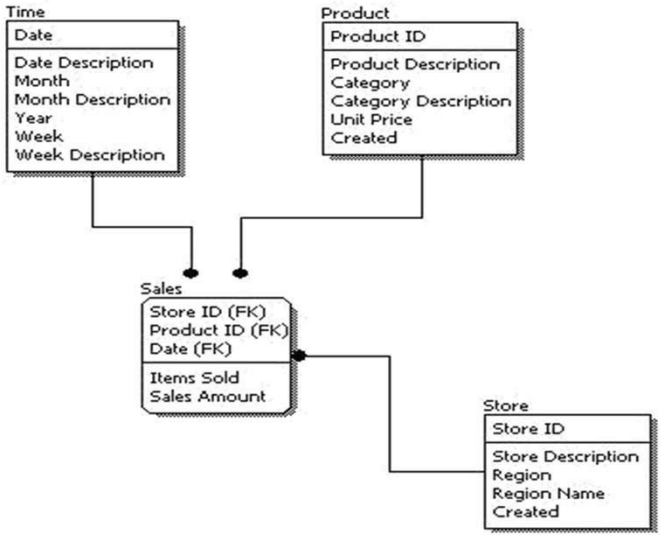
A relationship that is multi-valued in both directions is a many-to-many relationship. An employee can work on more than one project, and a project can have more than one employee. The questions "What does Dolores Quintana work on?", and "Who works on project IF1000?" both yield multiple answers. A many-to-many relationship can be expressed in a table with a column for each entity ("employees" and "projects").



Many-to-many relationships cannot be used in the data model because they cannot be represented by the relational model. Therefore, many-to-many relationships must be resolved early in the modelling process. The strategy for resolving many-to-many relationship is to replace the relationship with an association entity and then relate the two original entities to the association entity

4.5: Logical Model –an Example

Logical Model: Example



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

Exercise 1

- Scenario
  - Industry: Automobile manufacturing
  - Company: Millennium Motors
  - Value chain focus: Sales
- Sample business questions:
  - What are the top 10 selling car models this month?
  - How do this months top 10 selling models compare to the top 10 over the last six months?
  - Show me dealer sales by region by model by day
  - What is the total number of cars sold by month by dealer by state?
- List facts and dimensions

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

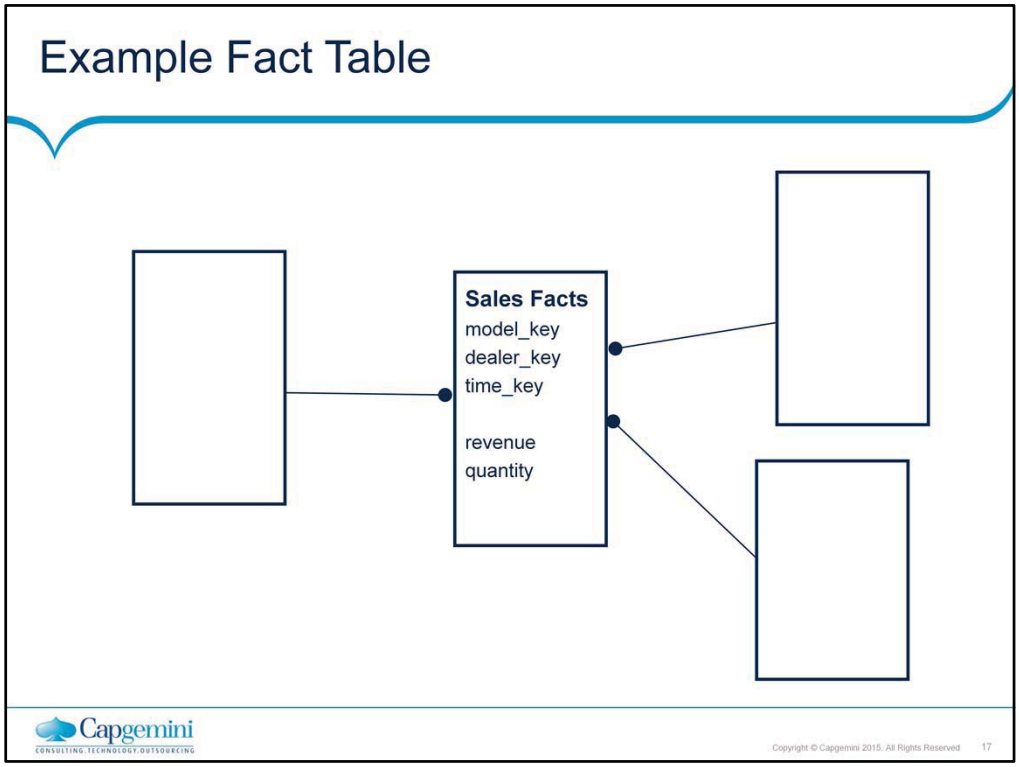
 15

Exercise 1 Solution

- Facts
  - Sales revenue
  - Quantity sold
- Dimensions
  - Model name
  - Month
  - Dealer name
  - Region
  - State
  - Date

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 16


Example Fact Table
Sales Facts
model_key
dealer_key
time_key
revenue
quantity
Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
17

Example Fact Table Records

Sales Facts

time_key	model_key	dealer_key	revenue	quantity
1	1	1	75840.27	2
1	2	1	152260.37	3
1	3	1	28360.15	1
1	4	1	132675.22	4
1	5	1	43789.45	1
1	1	2	35678.98	1
1	3	2	57864.78	2
1	5	2	92876.67	2

Primary Key

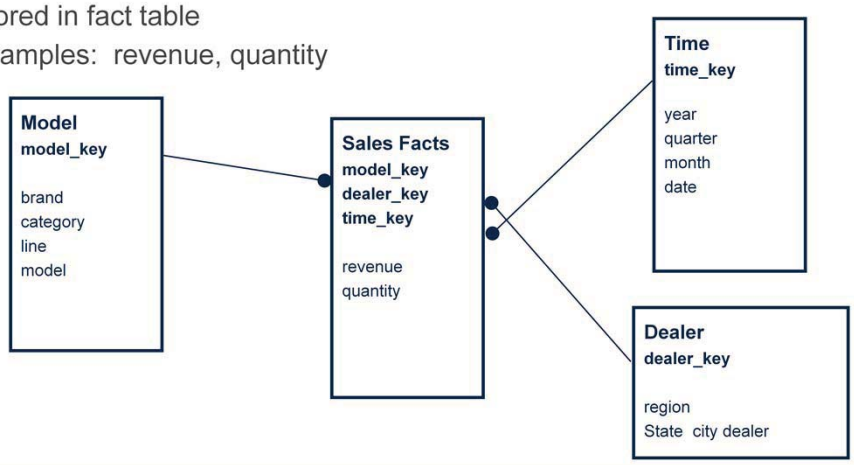
Facts

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 18

Facts

- Fully additive
  - Can be summed across any and all dimensions
  - Stored in fact table
  - Examples: revenue, quantity

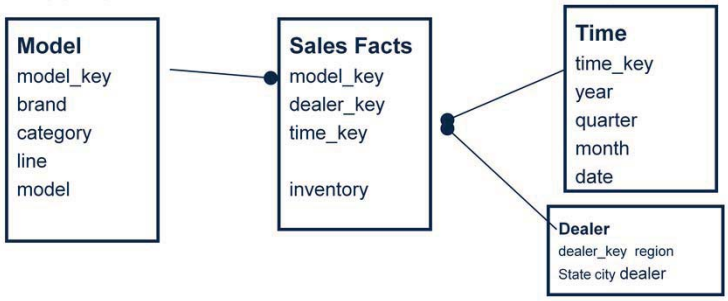


Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 19

Facts

- **Semi-additive**
  - Can be summed across most dimensions but not all
  - Examples: Inventory quantities, account balances, or personnel counts
  - Anything that measures a "level"
  - Must be careful with ad-hoc reporting
  - Often aggregated across the "forbidden dimension" by averaging

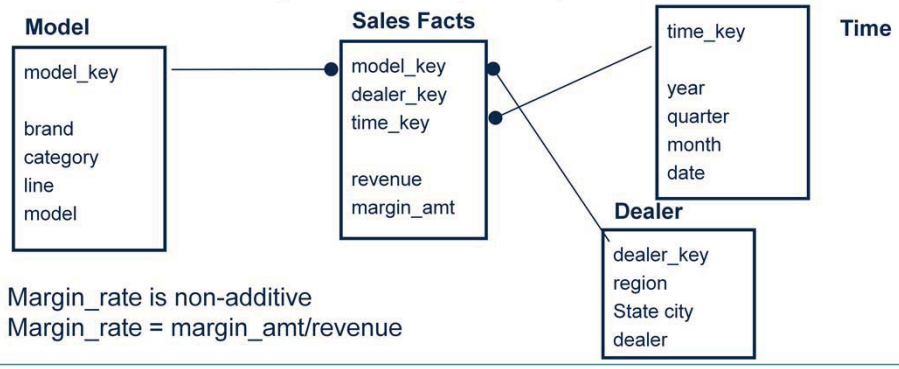


Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 20

Facts

- Non-Additive
  - Cannot be summed across any dimension
  - All ratios are non-additive
  - Break down to fully additive components, store them in fact table



Margin_rate is non-additive
Margin_rate = margin_amt/revenue

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

Unit Amounts

- Unit price, Unit cost, etc.
  - Are numeric, but not measures
  - Store the extended amounts which are additive
  - Unit amounts may be useful as dimensions for “price point analysis”
  - May store unit values to save space
- Factless Fact Table
  - A fact table with no measures in it
  - Nothing to measure...
  - except the convergence of dimensional attributes
  - Sometimes store a “1” for convenience
  - Examples: Attendance, Customer Assignments, Coverage

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

Example Dimension Table Records

Dealer Dimension

dealer_key	region	state	city	dealer
1	Northeast	Massachusetts	Boston	Honest Ted's
2	Northeast	Massachusetts	Boston	Stoller Co.
3	Southwest	Arizona	Tucson	Wright Motors
12	Southwest	California	San Diego	American
245	Central	Illinois	Chicago	Lugwig Motors

Synthetic Key

Attributes

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 23

Dimension Tables

- Characteristics
  - Hold the dimensional attributes
  - Usually have a large number of attributes (“wide”)
  - Add flags and indicators that make it easy to perform specific types of reports
  - Have small number of rows in comparison to fact tables (most of the time)
- Don’t normalize dimension table
  - Saves very little space
  - Impacts performance
  - Can confuse matters when multiple hierarchies exist
  - A star schema with normalized dimensions is called a "snowflake schema"
  - Usually advocated by software vendors whose product require snowflake for performance

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 24

4.6: Logical Model

Logical Data Design

Feature	Conceptual	Logical	Physical
Entity Names	✓	✓	
Entity Relationships	✓	✓	
Attributes		✓	
Primary Keys		✓	
Foreign Keys		✓	
Table Names			
Column Names			
Column Data Types			

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

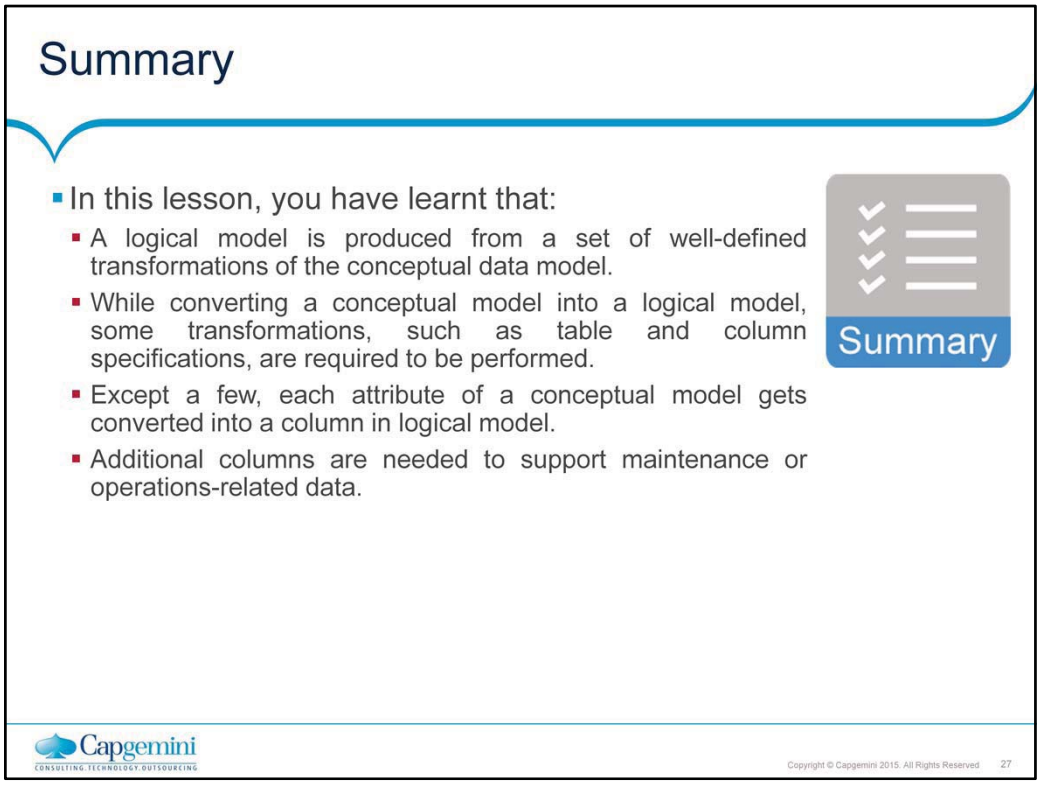
 25

Data modeling TOOLS

- ERwin Data Modeler
- Sybase PowerDesigner
- Database Workbench
- MagicDraw
- Microsoft SQL Server Management Studio

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 26



Add the notes here.

Review Question

- Question 1: The design of Logical Model is dependent on database.
  - True/False
- Question 2: Surrogate /Synthetic Keys are introduced in the Conceptual Model.
  - True/False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 28