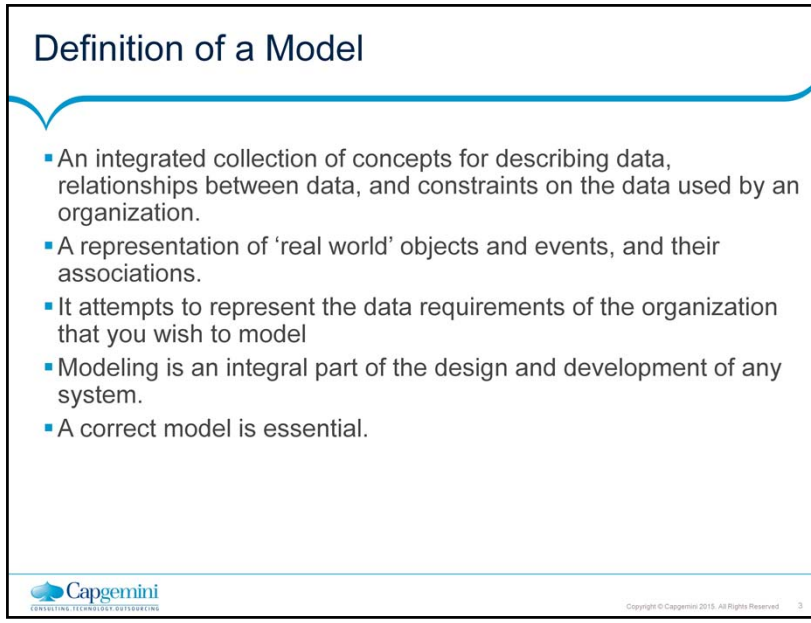
DBMS SQL

Lesson 07 : Introduction to Data Modeling E-R Model and Normalization

Table Of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING



What is a model?

A model serves two primary purposes:

As a true representation of some aspects of the real world, a model enables clearer communication about those aspects.

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.

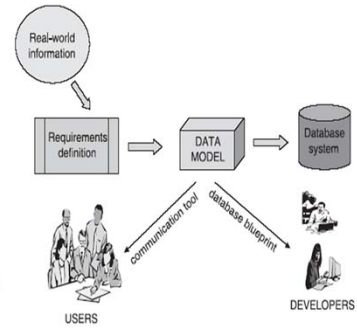
2) It enables the database practitioners to implement the database system exactly conforming to the information requirements.

A data model, therefore, serves as a critical tool for communication with the users and it also serves as a blueprint of the database system for the developers.

Without a proper data model, an adequate database system cannot be correctly designed and implemented. A good data model forms an essential prerequisite for any successful database system. Unless the data modelers represent the information requirements of the organization in a proper data model, the database design will be totally ineffective.

What is Data Modeling?

- Data modeling is a technique for exploring the data structures needed to support an organization's information need.
- It would be a conceptual representation or a replica of the data structure required in the database system.
- A data model focuses on which data is required and how the data should be organized.
- At the conceptual level, the data model is independent of any hardware or software constraints.



Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

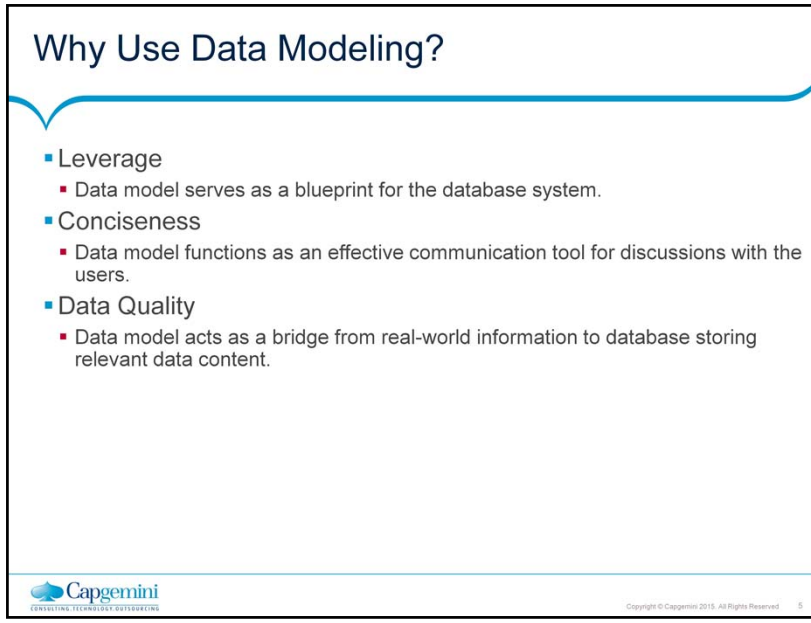
 4

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.

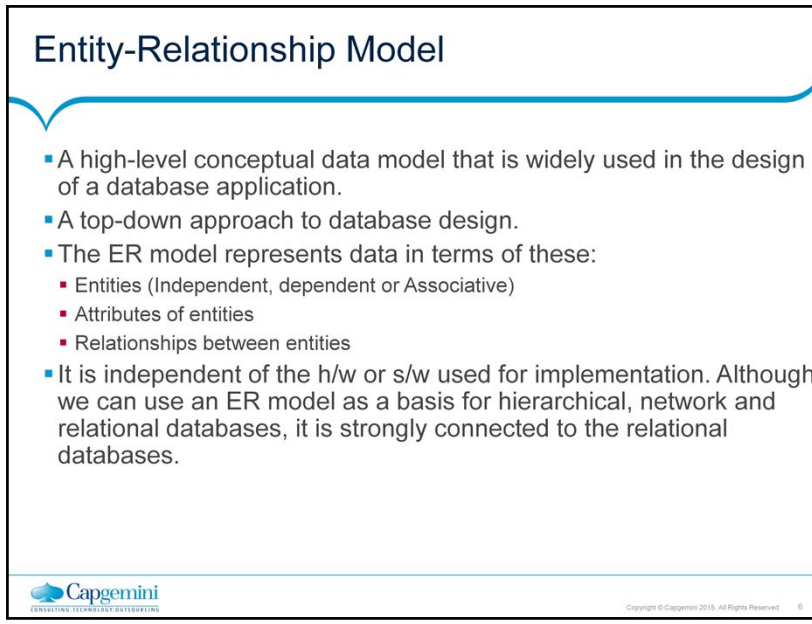


Why Use Data Modeling?

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.

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.

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.



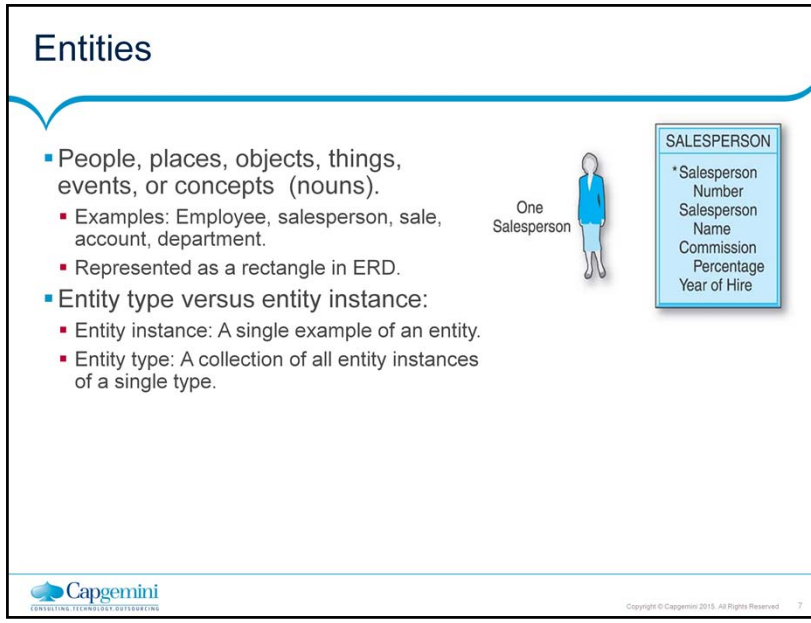
What is Entity-Relationship Model?

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.

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.

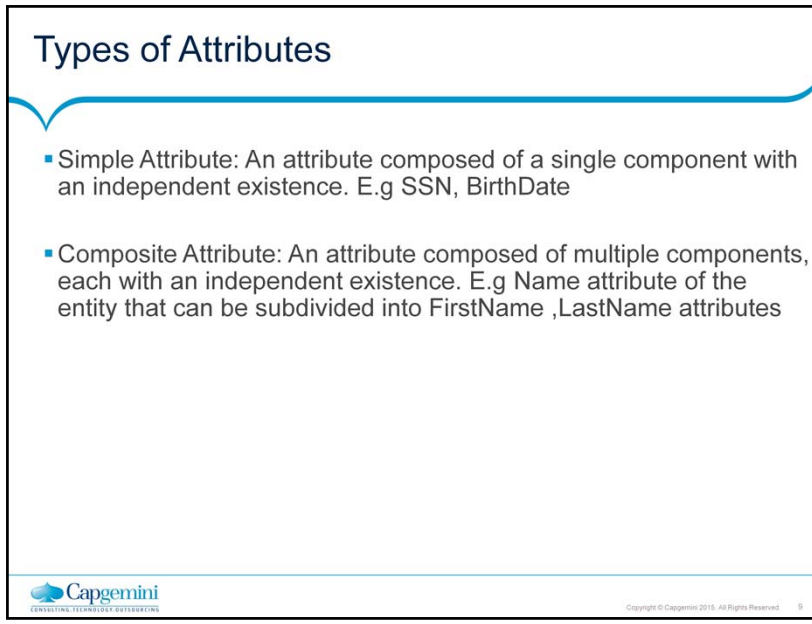


The basic concept of the ER model is an **entity**, which represents a set of
objects in the 'real world' that share the same properties. Each object, which should be uniquely identifiable within the set, is called an **entity occurrence**. An entity has an independent existence and can represent objects with a physical (or 'real') existence or objects with a conceptual (or 'abstract') existence

Attributes

- Attributes describe the entity with which they are associated.
- They are characteristics of an Entity.

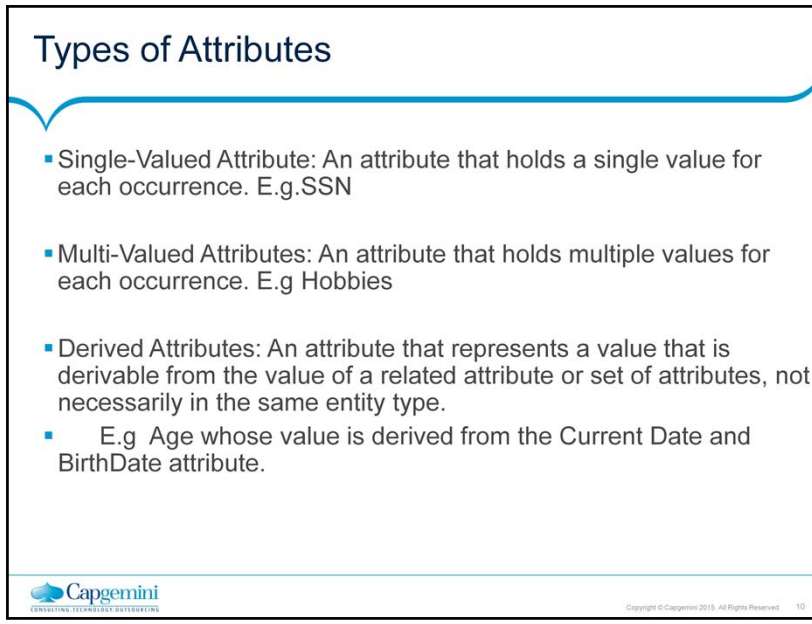
Eg: Customer Name is an attribute of the entity Customer.



Attributes

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.



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.

Relationship

- A relationship represents an association between two or more entities. An example of a relationship would be as follows:
  - Employees are assigned to projects.
  - Projects have subtasks.
  - Departments manage one or more projects.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

Key Types

Superkey:

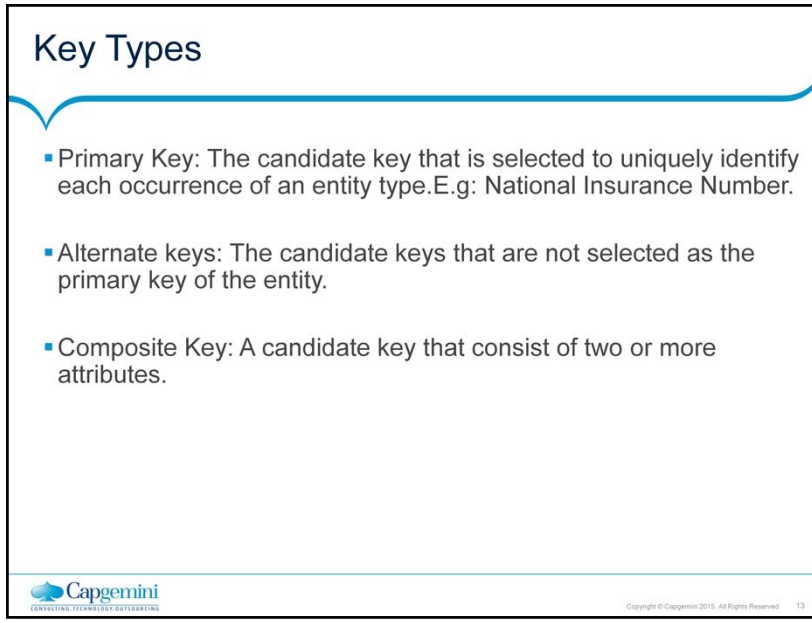
- An attribute, or set of attributes, that uniquely identifies each entity occurrence.
- If we add additional attributes to a primary key, the resulting combination would still uniquely identify an instance of the entity set. Such augmented keys are called superkey

Candidate Key:

- A superkey that contains only the minimum number of attributes necessary for unique identification of each entity occurrence. Ex. BranchNo in entity Branch.
- The minimal super key is also referred as candidate key

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12



Primary Keys: The candidate key that is selected to identify each entity occurrence.

They are a strong concept that is usually enforced for every table.

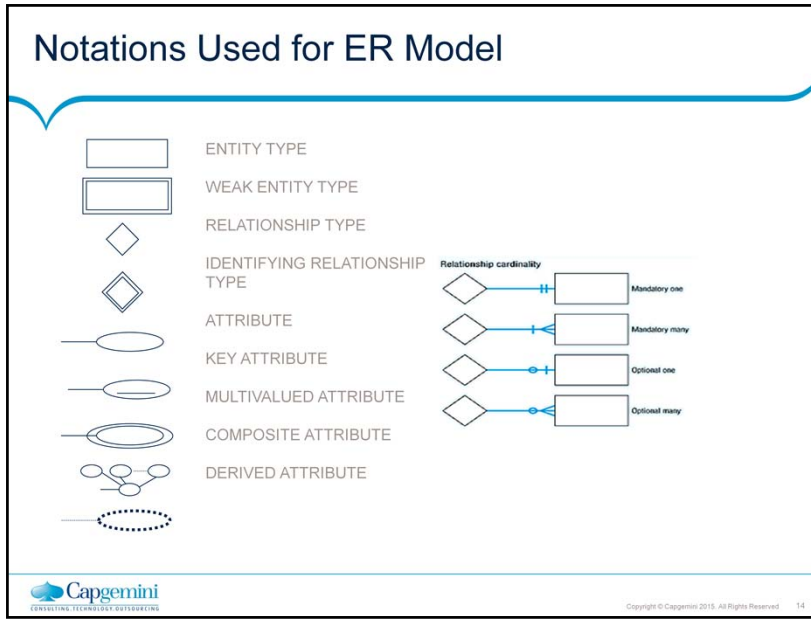
- They can be made up of one or more columns; each has to be mandatory.
- They are declarative as a constraint and can be named. When creating a primary key constraint, Oracle automatically creates a unique index in association with it.
- A foreign key usually refers to the primary key of a table, but may also refer to a unique key.

Tables that do not have a primary key should have a unique key.

Unique Keys: A unique key is a key that for some reason was not selected to be the primary key. The reasons may have been:

- Allowed nulls. Nulls may be allowed in Unique keys columns.
- Updatable. Unique key values may change but still need to remain unique. For example, the home phone number of an employee or the license plate for a car.

There may be more than one unique key for each table.



Notations Used for ER Model:

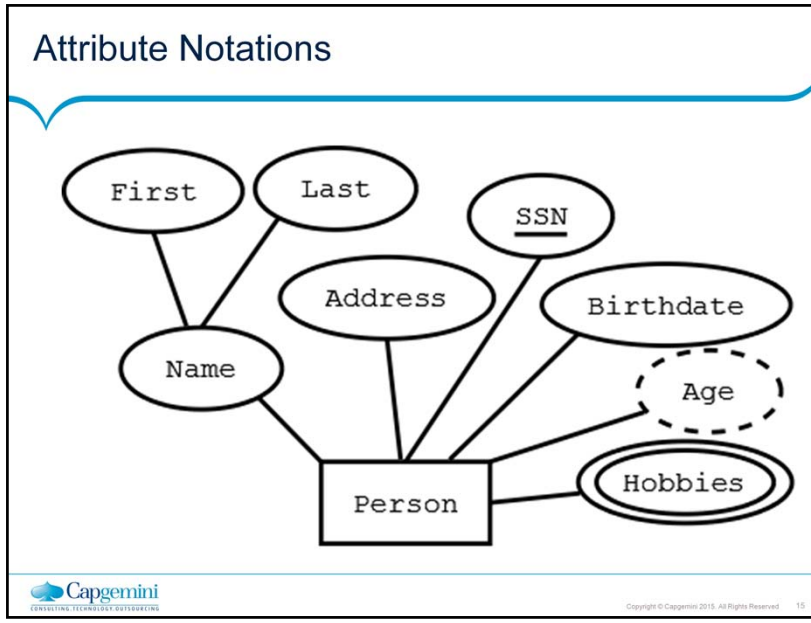
Entities are represented by labeled rectangles. The label is the name of the entity. Entity names should be singular nouns.

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.

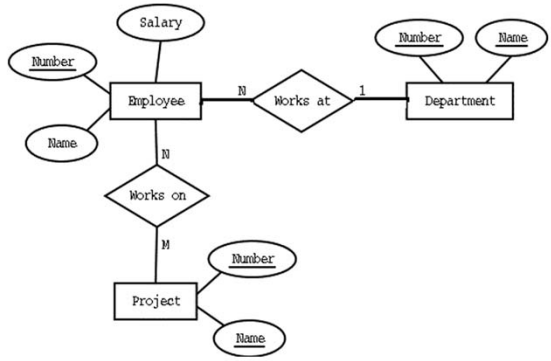
Cardinality of many is represented by a line ending in a crow's foot. If the crow's foot is omitted, the cardinality is one.

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.



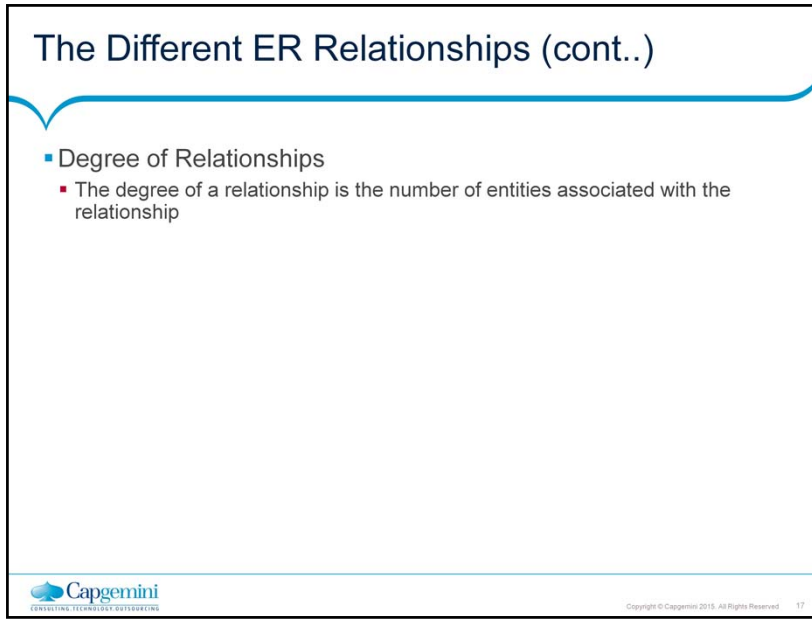
Rectangle -- Entity
Ellipses -- Attribute (underlined attributes are [part of] the primary key)
Double ellipses -- multi-valued attribute
Dashed ellipses-- derived attribute, e.g. age is derivable from birthdate and current date.

The E-R Model - An Example



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

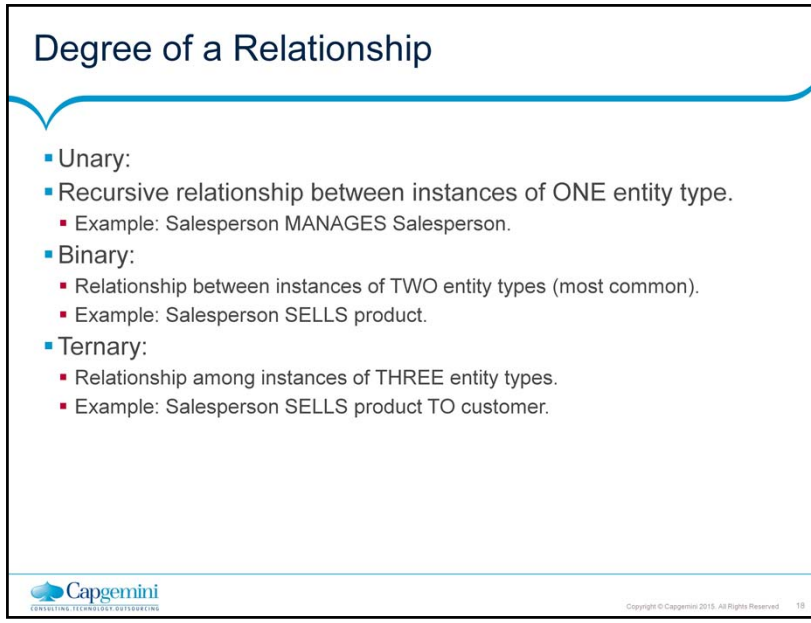


Degree of Relationship

The degree of a relationship is the number of entities associated with the relationship. The n-ary relationship is the general form for degree n. Special cases are the binary and ternary, with the degree 2 and 3 respectively.

Binary relationship is the most common type of association in the real world and is defined between two entities. A recursive binary relationship occurs when an entity is related to itself. An example might be "some employees are married to other employees".

A ternary relationship involves three entities and is used when a binary relationship is inadequate. Many modeling approaches recognize only binary relationships. Ternary or n-ary relationships are disintegrated into two or more binary relationships.



The entities involved in a particular relationship are referred to as participants. The number of participants in a relationship is called the **degree** and indicates the number of entities involved in a relationship. A relationship of degree one is called **unary**, which is commonly referred to as a *recursive* relationship. A relationship of degree two is called **binary**. The two relationships shown. A relationship of a degree higher than binary is called a **complex relationship**. A relationship of degree three is called **ternary**. A relationship of degree four is called **quaternary**, and a relationship of a higher degree is called ***n*-ary**. The most popular type of relationship you'll come across is binary, but occasionally you'll come across unary or ternary, and less frequently quaternary.

Types of Binary Relationships

- One-to-one (1:1) binary relationship:
  - One instance of an entity type is related to only one instance of another entity type, and vice versa.
    - Line 3.1.1
  - Example: A salesperson can have only ONE office; an office can be assigned to only ONE salesperson.
- One-to-many (1:N) binary relationship:
  - One instance of an entity type (the “one” side) is related to many instances of another entity type (the “many” side), but an entity instance on the “many” side can be related to only one instance on the “one” side.
  - Example: A salesperson can have MANY sales transaction, but one sales transaction can only be done by ONE salesperson.

Types of Binary Relationships

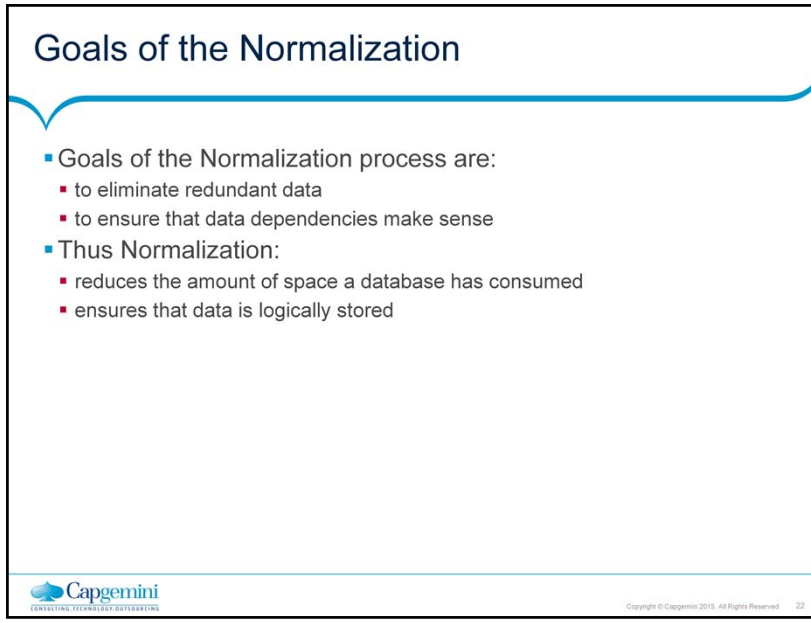
- Many-to-many (M:N) binary relationship:
  - Instances of each entity type can be related to many instances of the other entity type.
  - Example: A salesperson can sell MANY products, and a product can be sold by MANY salespersons.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

Normalization

- Normalization is a technique for designing relational database tables.
- Normalization is used:
  - to minimize duplication of information
  - to safeguard the database against certain types of problems
- Thus, Normalization is a process of efficiently organizing data in a database.
- Normalization is done within the framework of five normal forms(numbered from first to fifth).
- Most designs conform to at least the third normal form.
- Normalization rules are designed to prevent anomalies and data inconsistencies.



Normalization (contd.):

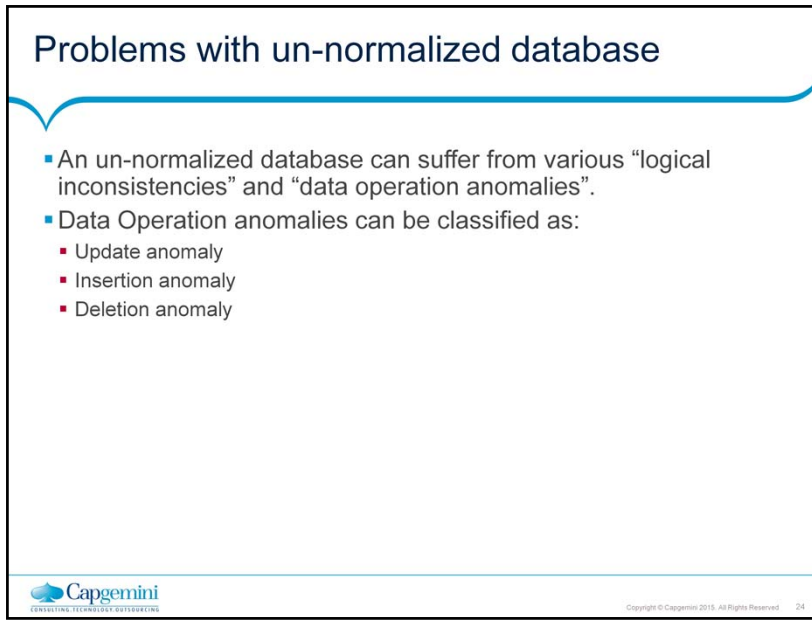
- Higher degrees of normalization typically involve more tables and create the need for a larger number of joins, which can reduce performance.
- Accordingly:
  - More highly normalized tables are typically used in database applications involving many isolated transactions
    For example: an Automated teller machine
  - While less normalized tables tend to be used in database applications that need to map complex relationships between data entities and data attributes
    For example: a reporting application, or a full-text search application
- Database theory describes a table's degree of normalization in terms of Normal Forms of successively higher degrees of strictness.
  For example: A table in Third Normal form (3NF), is consequently in Second Normal form (2NF) as well but the reverse is not necessarily true.
- Although the Normal Forms are often defined informally in terms of the characteristics of tables, rigorous definitions of the Normal Forms are concerned with the characteristics of mathematical constructs known as "Relations". Whenever information is represented relationally, it is meaningful to consider the extent to which the representation is normalized.

Benefits of Normalization

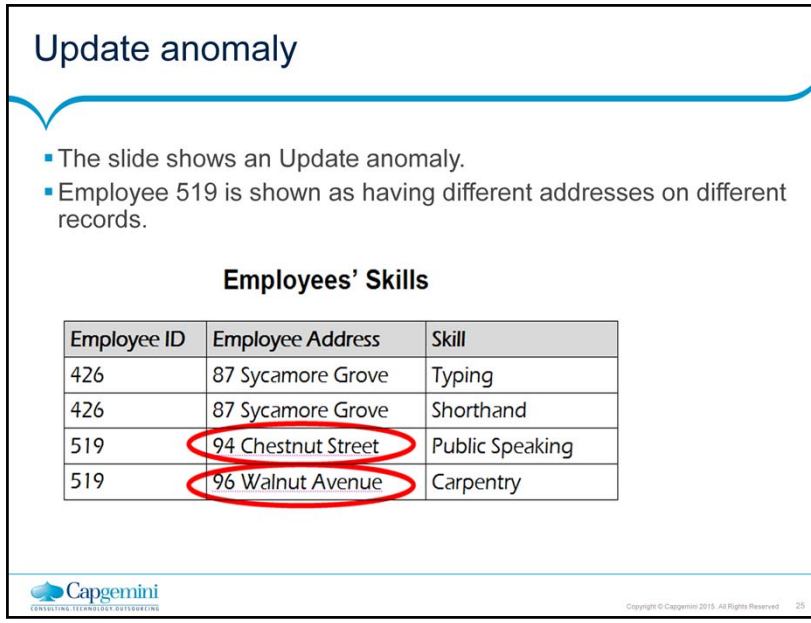
- Benefits of Normalization are given below:
  - Greater overall database organization
  - Reduction of redundant data
  - Data consistency within the database
  - A much more flexible database design
  - A better handle on database security

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

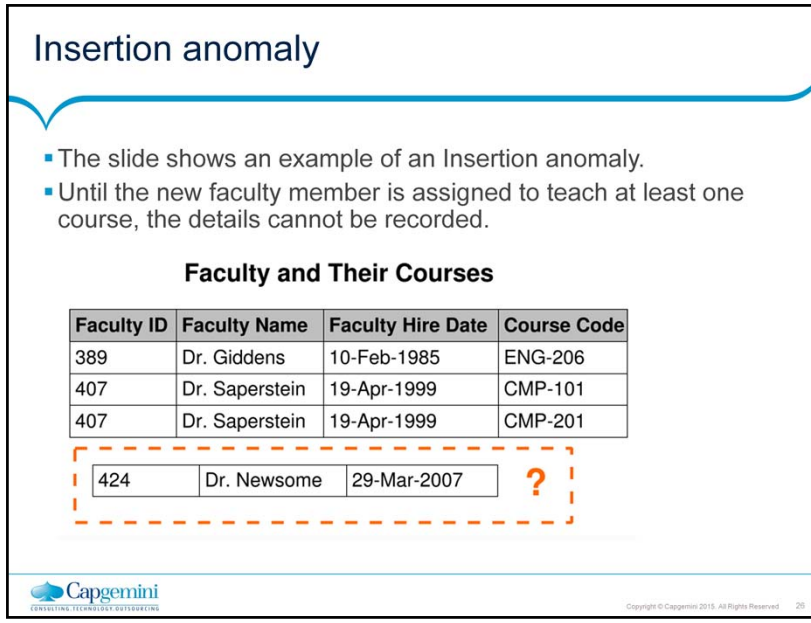
 23



The purpose of Normalization is to ensure:
there is no duplication of data
there is no unnecessary data stored within a Database
all the attributes (columns) of a table are completely dependent on the primary key of a table only, and not on any other attribute.

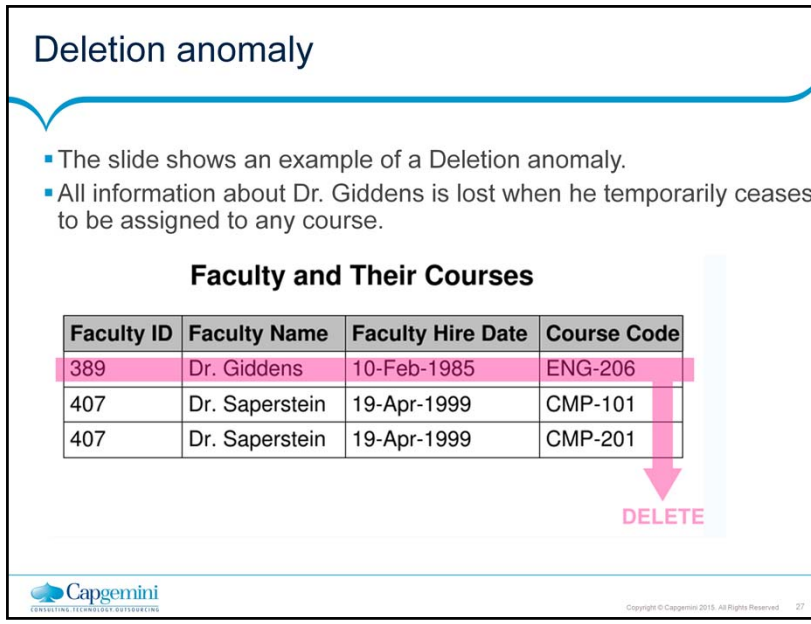


Update anomaly:
The same information can be expressed on multiple records. Therefore updates to the table may result in logical inconsistencies.
For example: Each record in an "Employees' Skills" table might contain an Employee ID, Employee Address, and Skill. Thus a change of address for a particular employee will potentially need to be applied to multiple records (one for each of his skills). If the update is not carried through successfully — that is, if the employee's address is updated on some records but not on others — then the table is left in an inconsistent state. Specifically, the table provides conflicting answers to the question of what is the address of a particular employee. This phenomenon is known as an "Update anomaly".
The slide shows an Update anomaly. Employee 519 is shown as having different addresses on different records.



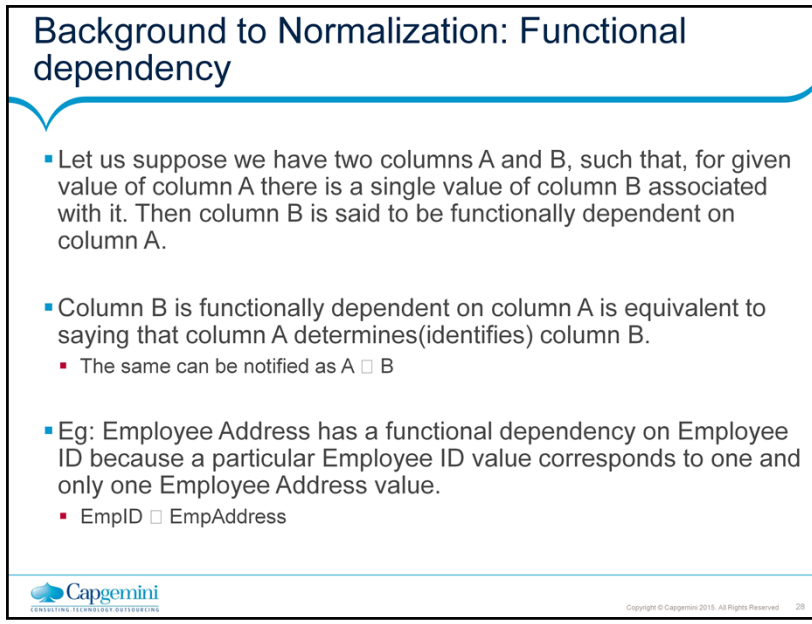
Insertion anomaly:
There are circumstances in which certain facts cannot be recorded at all.

For example: Each record in “Faculty and Their Courses” table might contain a Faculty ID, Faculty Name, Faculty Hire Date, and Course Code — thus we can record the details of any faculty member who teaches at least one course. However, we cannot record the details of a newly-hired faculty member who has not yet been assigned to teach any courses. This phenomenon is known as an “Insertion anomaly”.
The slide shows an Insertion anomaly. Until the new faculty member is assigned to teach at least one course, his details cannot be recorded.



Deletion Anomaly:

- There are circumstances in which the deletion of data representing certain facts necessitates the deletion of data representing completely different facts. The “Faculty and Their Courses” table described in the previous example suffers from this type of anomaly. If a faculty member temporarily ceases to be assigned to any course, we must delete the last of the records on which that faculty member is displayed. This phenomenon is known as a “Deletion anomaly”.
- The slide shows a Deletion anomaly. All information about Dr. Giddens is lost when he temporarily ceases to be assigned to any courses.



Note that the reverse need not be true. Several employees can live at the same address and therefore one Employee Address value can correspond to more than one Employee ID. Employee ID is therefore not functionally dependent on Employee Address.

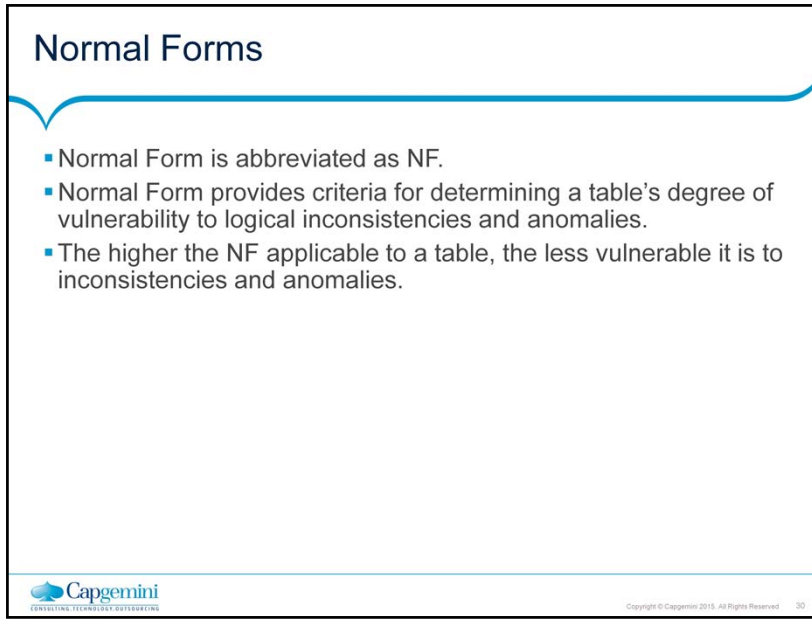
An attribute may be functionally dependent either on a single attribute or on a combination of attributes. It is not possible to determine the extent to which a design is normalized without understanding what functional dependencies apply to the attributes within its tables. Understanding this concept, in turn, requires knowledge of the problem domain.

Background to Normalization (contd.):

Three Types of Functional Dependencies

- **Full Dependency:** In a relation, the attribute(s) B is fully functional dependent on A, if B is functionally dependent on A but not on any proper subset of A.
- **Partial Dependency:** It is a relation, where there is some attribute that can be removed from A and the dependency still holds.
  For example: Staff_No, Sname → Branch_No
- **Transitive Dependency:** In a relation, if attribute(s) A→B and B→C, then C is transitively dependent on A via B (provided that A is not functionally dependent on B or C)
  For example: Staff_No → Branch_No, and Branch_No → BAddress
  For example: {Employee Address} has a functional dependency on {Employee ID, Skill}, but not a full functional dependency, because it is also dependent on {Employee ID}.

- **Trivial functional dependency:** A trivial functional dependency is a functional dependency of an attribute on a superset of itself. {Employee ID, Employee Address} → {Employee Address} is trivial, as is {Employee Address} → {Employee Address}.
- **Transitive dependency:** A transitive dependency is an indirect functional dependency, one in which X → Z only by virtue of X→Y and Y→Z.
- **Multivalued dependency:** A multi-valued dependency is a constraint according to which the presence of certain rows in a table implies the presence of certain other rows. Refer the Multivalued Dependency article for a rigorous definition.
- **Join dependency:** A table T is subject to a join dependency if T can always be recreated by joining multiple tables each having a subset of the attributes of T.
- **Superkey:** A super key is an attribute or set of attributes that uniquely identifies rows within a table. In other words, two distinct rows are always guaranteed to have distinct superkeys. {Employee ID, Employee Address, Skill} will be a superkey for the "Employees' Skills" table; {Employee ID, Skill} will also be a superkey.
- **Candidate key:** A candidate key is a minimal superkey, that is a superkey for which we can say that no proper subset of it is also a superkey. {Employee Id, Skill} will be a candidate key for the "Employees' Skills" table.
- **Non-prime attribute:** A non-prime attribute is an attribute that does not occur in any candidate key. Employee Address will be a non-prime attribute in the "Employees' Skills" table.
- **Primary key:** Most DBMSs require a table to be defined as having a single unique key, rather than a number of possible unique keys. A primary key is a key which the database designer has designated for this purpose.



Normal Forms:

- The Normal Forms (abbrev. NF) of relational database theory provide criteria for determining a table's degree of vulnerability to logical inconsistencies and anomalies. The higher the Normal Form applicable to a table, the less vulnerable it is to inconsistencies and anomalies.
- Each table has a "highest normal form" (HNF). By definition, a table always meets the requirements of its HNF and of all normal forms lower than its HNF. Again by definition, a table fails to meet the requirements of any Normal Form higher than its HNF.
- The Normal Forms are applicable to individual tables. To say that an entire database is in normal form n implies that all its tables are in normal form n.
- Newcomers to database design sometimes suppose that normalization proceeds in an iterative fashion, i.e. a 1NF design are first normalized to 2NF, then to 3NF, and so on. This is not an accurate description of how normalization typically works. A sensibly designed table is likely to be in 3NF on the first attempt. Furthermore, if it is 3NF, it is overwhelmingly likely to have an HNF of 5NF. Achieving the "higher" normal forms (above 3NF) does not usually require an extra expenditure of effort on the part of the designer, because 3NF tables usually need no modification to meet the requirements of these higher normal forms.
- Edgar F. Codd originally defined the first three normal forms (1NF, 2NF, and 3NF). These normal forms have been summarized as requiring that all non-key attributes be dependent on "the key, the whole key and nothing but the key". The fourth and fifth normal forms (4NF and 5NF) deal specifically with the representation of many-to-many and one-to-many relationships among attributes. Sixth normal form (6NF) incorporates considerations relevant to temporal databases.

Un-normalized Relation

- An Unnormalized NF relation is a table that contains one or more Repeating Groups or Non-Atomic values.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 31

Case Study: Normalization

- Given an EmpProject Table structure
- Note: EmpProject is an un-normalized relation

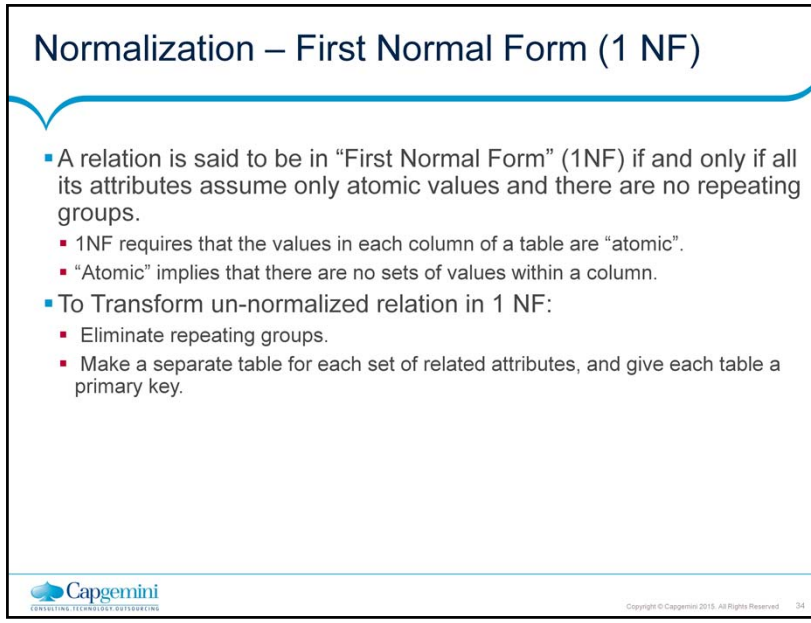
Proj Code	Proj Type	Proj Desc	Empno	Ename	Grade	Sal scale	Proj Join Date	Alloc Time
001	APP	LNG	46	JONES	A1	5	12/1/1998	24
001	APP	LNG	92	SMITH	A2	4	2/1/1999	24
001	APP	LNG	96	BLACK	B1	9	2/1/1999	18
004	MAI	SHO	72	JACK	A2	4	2/4/1999	6
004	MAI	SHO	92	SMITH	A2	4	5/5/1999	6

Normalization

- Problems with the above kind of implementation
  - Data Redundancy : Proj. type and description are unnecessarily repeated many times.
  - An employee can't logically exist.
  - If project type changes many records will require updating in an identical manner.
  - If employee Smith gains promotion, many records will require updating.
  - If management decides to change the relationship between the grade and salary scale then many updates will be required.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 33



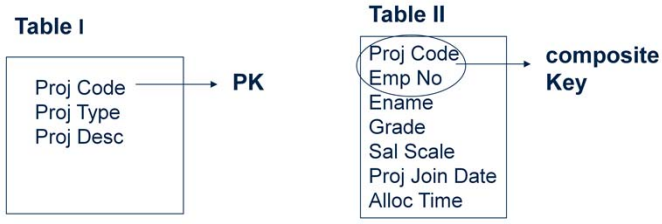
First Normal Form (1NF):

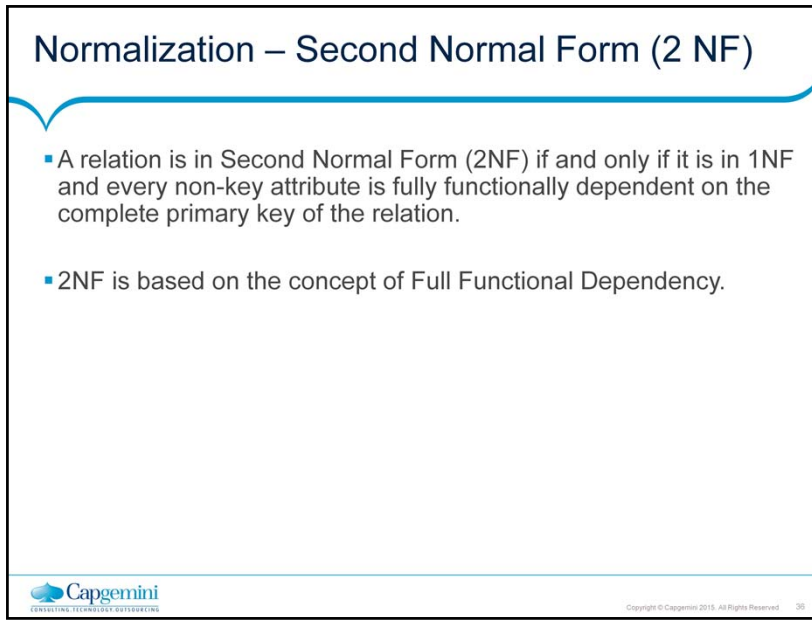
- A table is in first normal form (1NF) if and only if it faithfully represents a relation. Given that database tables embody a relation-like form, the defining characteristic of one in First Normal form is that it does not allow duplicate rows or nulls. To put it simply, a table with a unique key (which, by definition, prevents duplicate rows) and without any nullable columns is in 1NF.
- **Note:**
  - The restriction on nullable columns as a 1NF requirement, as espoused by Chris Date, et. al., is controversial. This particular requirement for 1NF is a direct contradiction to Dr. Codd’s vision of the relational database, in which he stated that “null values” must be supported in a fully relational DBMS in order to represent “missing information and inapplicable information in a systematic way, independent of data type”.
  - By redefining 1NF to exclude nullable columns in 1NF, no level of normalization can ever be achieved unless all nullable columns are completely eliminated from the entire database. This is in line with Date’s and Darwen’s vision of the perfect relational database, however it can introduce additional complexities in SQL databases to the point of impracticality.

contd.

Normalization – First Normal Form (1 NF)

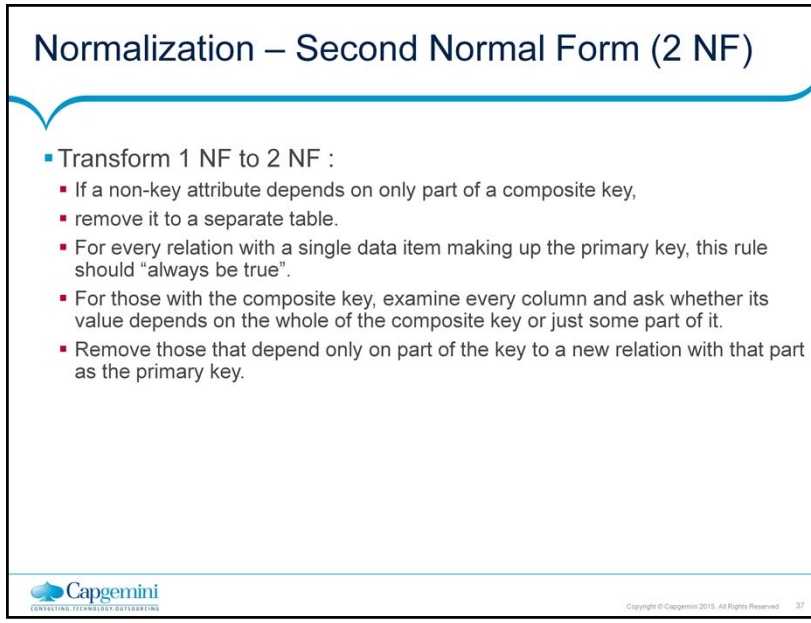
- Rule:
  - A relation is in the first normal form if the intersection of any column and row contains only one value
  - Identify any suitable primary key
  - Remove repeating groups to simplify relationship that may lead to multiple table design





Second Normal Form (2NF):

- Where the First Normal Form deals with atomicity of data, the Second Normal Form (or 2NF) deals with relationships between composite key columns and non-key columns. As stated earlier, the normal forms are progressive, so to achieve Second Normal form, your tables must already be in First Normal Form.
- In case of the Second Normal form (or 2NF), any non-key columns must depend on the entire primary key. In the case of a composite primary key, this means that a non-key column cannot depend on only part of the composite key.



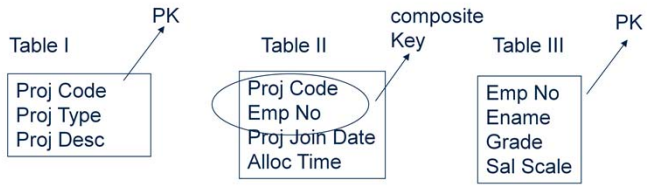
Rules for 2NF:

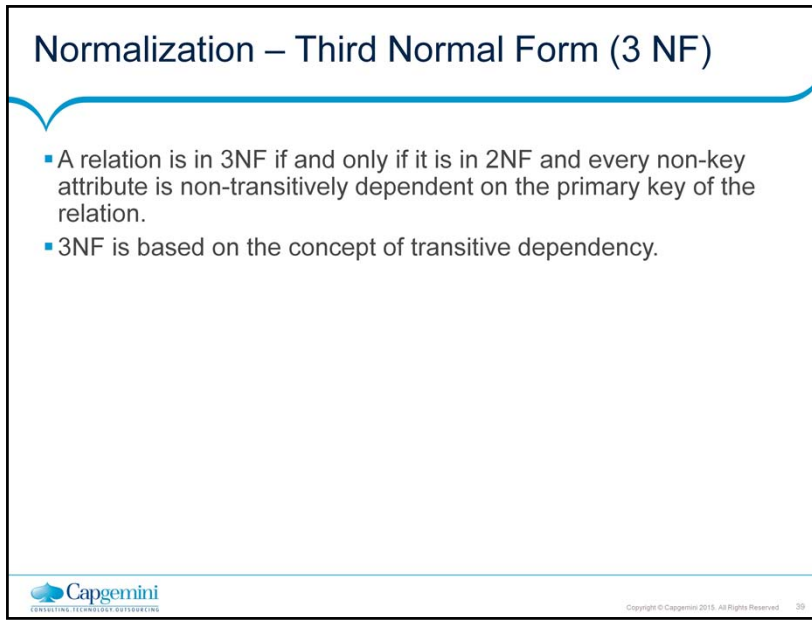
The Second Normal Form involves the idea of **Functional Dependency**.

- **Functional Dependency:** Attribute B has a functional dependency on attribute A, if for each value of attribute A, there is exactly one value of attribute B.
  - **For example:** In an Emp table, Employee Address has a functional dependency on Employee ID because a particular Employee ID value corresponds to one and only one Employee Address value.
  - Note that the reverse need not be true.
    For example: Several employees can live at the same address, and therefore one Employee Address value can correspond to more than one Employee ID. Employee ID is therefore not functionally dependent on Employee Address.
  - An attribute may be functionally dependent either on a single attribute or on a combination of attributes. It is not possible to determine "the extent to which a design is normalized" without understanding the "functional dependencies" that apply to the attributes within its tables.
  - Understanding this concept, in turn, requires knowledge of the problem domain.
    For example: An Employer may require certain employees to split their time between two locations, such as New York City and London, and therefore want to allow Employees to have more than one Employee Address. In this case, Employee Address will no longer be functionally dependent on Employee ID.

contd.

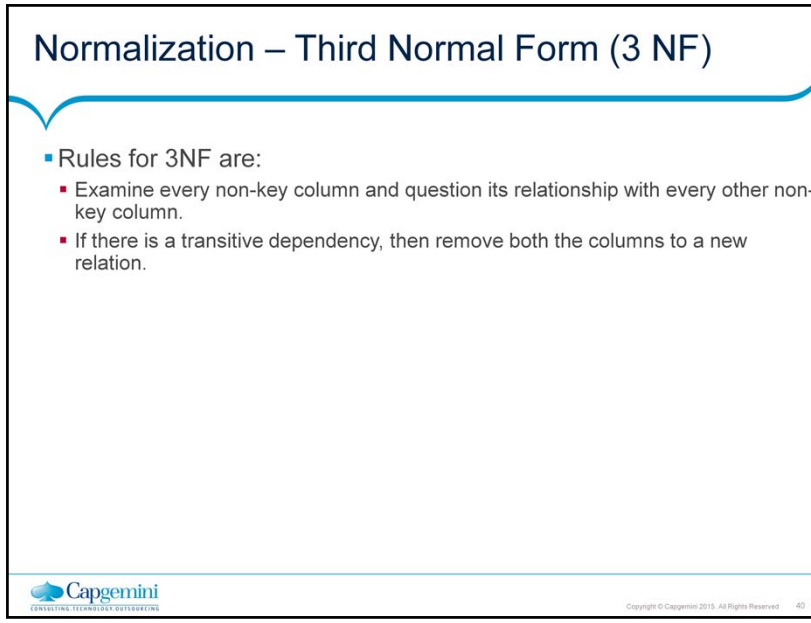
Normalization – Second Normal Form (2 NF)





Third Normal Form (3NF):

- Third Normal Form (3NF) requires that all columns depend directly on the primary key. Tables violate the Third Normal Form when one column depends on another column, which in turn depends on the primary key (a transitive dependency).
- One way to identify transitive dependencies is to look at your table, and see if any columns require updating if another column in the table is updated. If such a column exists, it probably violates 3NF.
- Third Normal Form: Eliminate Data Not Dependent On Key



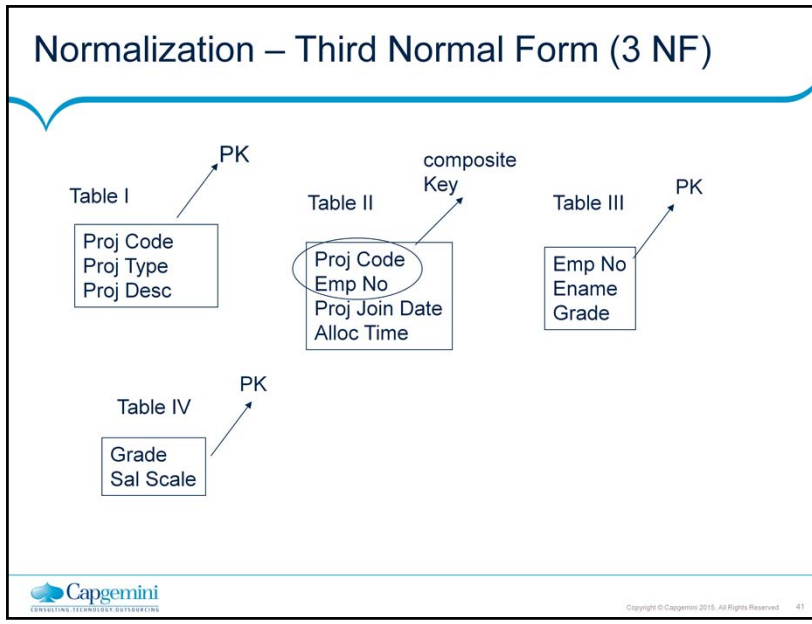
Criteria for 3NF:

- The table must be in 2NF.
- Every non-prime attribute of the table must be non-transitively dependent on each candidate key. A violation of 3NF will mean that at least one non-prime attribute is only indirectly dependent (transitively dependent) on a candidate key.
- **For example:**

➢ Consider a “Departments” table whose attributes are Department ID, Department Name, Manager ID, and Manager Hire Date. And suppose that each manager can manage one or more departments. {Department ID} is a candidate key.

➢ Although Manager Hire Date is functionally dependent on the candidate key {Department ID}, this is only because Manager Hire Date depends on Manager ID, which in turn depends on Department ID.

➢ This transitive dependency means the table is not in 3NF unless the manager can be hired for more than one department and the date represents the hiring date of that manager only for this department.


Normalization – Third Normal Form (3 NF)
Table I
PK
Proj Code
Proj Type
Proj Desc
Table II
composite Key
Proj Code
Emp No
Proj Join Date
Alloc Time
Table III
PK
Emp No
Ename
Grade
Table IV
PK
Grade
Sal Scale
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
41

Normalization Summarization

- 1 NF- Ensure all values are atomic and Eliminate Repeating Groups
- 2 NF- Eliminate Partial Dependencies
- 3 NF- Eliminate Transitive Dependencies

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 42

Drawbacks of Normalization

- Typically, in a normalized database, more joins are required to pull together information from multiple tables.
- Joins require additional I/O to process, and are therefore more expensive from a performance standpoint than single-table lookups.
- Additionally, a normalized database often incurs additional CPU processing. CPU resources are required to perform join logic and to maintain data and referential integrity.

Summary

- In this lesson, you have learnt that:
  - Data Modeling
  - E-R Modeling
  - Normalization
    - Problems with un-normalized database
    - Normal Forms
    - Benefits of Normalization
    - Drawbacks of Normalization

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Review Question

- Question 1: The higher the NF applicable to a table, the less vulnerable it is to inconsistencies and anomalies.
  - True/False
- Question 2: ____remove partial key dependencies

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

