# DBMS SQL

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

Capgemini

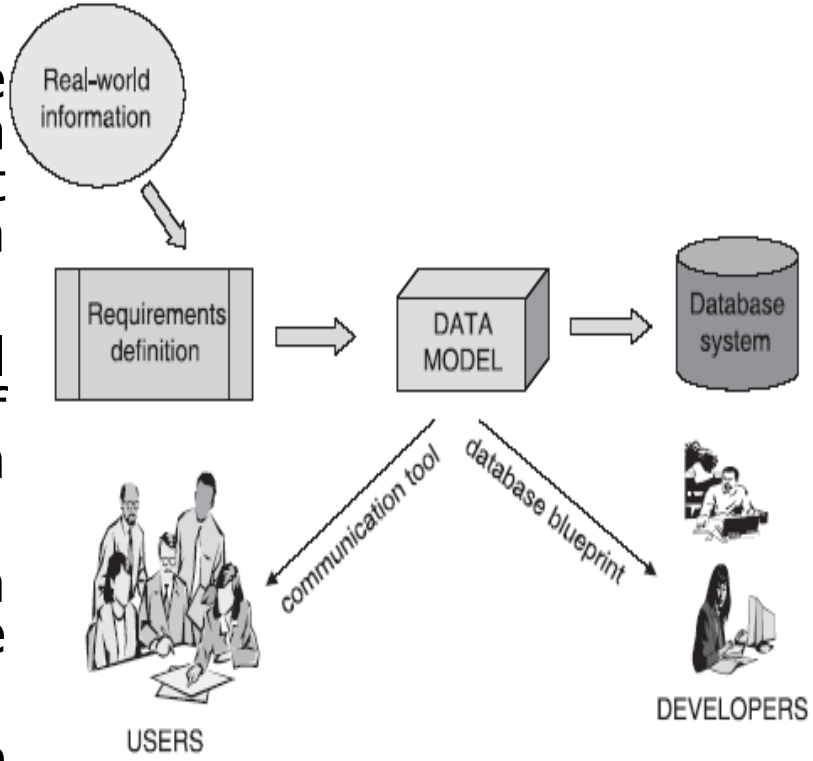
# Table Of Contents

# Definition of a Model

- An integrated collection of concepts for describing data, relationships between data, and constraints on the data used by an organization.
- A representation of ‘real world’ objects and events, and their associations.
- It attempts to represent the data requirements of the organization that you wish to model
- Modeling is an integral part of the design and development of any system.
- A correct model is essential.

# 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.



# Why Use Data Modeling?

- Leverage
  - Data model serves as a blueprint for the database system.
- Conciseness
  - Data model functions as an effective communication tool for discussions with the users.
- Data Quality
  - Data model acts as a bridge from real-world information to database storing relevant data content.

# Entity-Relationship Model

- A high-level conceptual data model that is widely used in the design of a database application.
- A top-down approach to database design.
- The ER model represents data in terms of these:
  - Entities (Independent, dependent or Associative)
  - Attributes of entities
  - Relationships between entities
- It is independent of the h/w or s/w used for implementation. Although we can use an ER model as a basis for hierarchical, network and relational databases, it is strongly connected to the relational databases.

# Entities

- People, places, objects, things, events, or concepts (nouns).
  - Examples: Employee, salesperson, sale, account, department.
  - Represented as a rectangle in ERD.
- Entity type versus entity instance:
  - Entity instance: A single example of an entity.
  - Entity type: A collection of all entity instances of a single type.


# 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.

# Types of Attributes

- Simple Attribute: An attribute composed of a single component with an independent existence. E.g SSN, BirthDate

- Composite Attribute: An attribute composed of multiple components, each with an independent existence. E.g Name attribute of the entity that can be subdivided into FirstName ,LastName attributes

# Types of Attributes

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

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

- 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 Age whose value is derived from the Current Date and BirthDate attribute.

# 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.

# 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

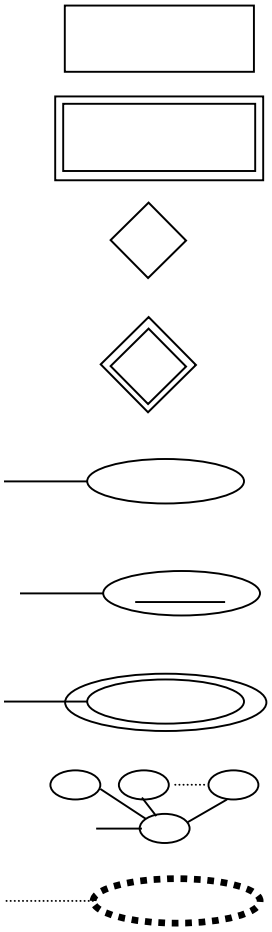
# Key Types

- Primary Key: The candidate key that is selected to uniquely identify each occurrence of an entity type.E.g: National Insurance Number.

- Alternate keys: The candidate keys that are not selected as the primary key of the entity.

- Composite Key: A candidate key that consist of two or more attributes.

# Notations Used for ER Model



ENTITY TYPE

WEAK ENTITY TYPE

RELATIONSHIP TYPE

IDENTIFYING RELATIONSHIP TYPE

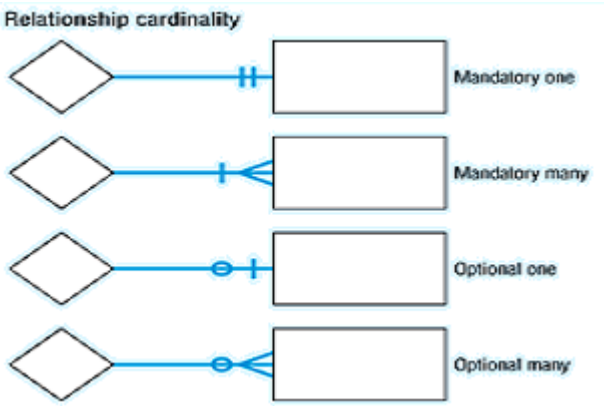
ATTRIBUTE

KEY ATTRIBUTE

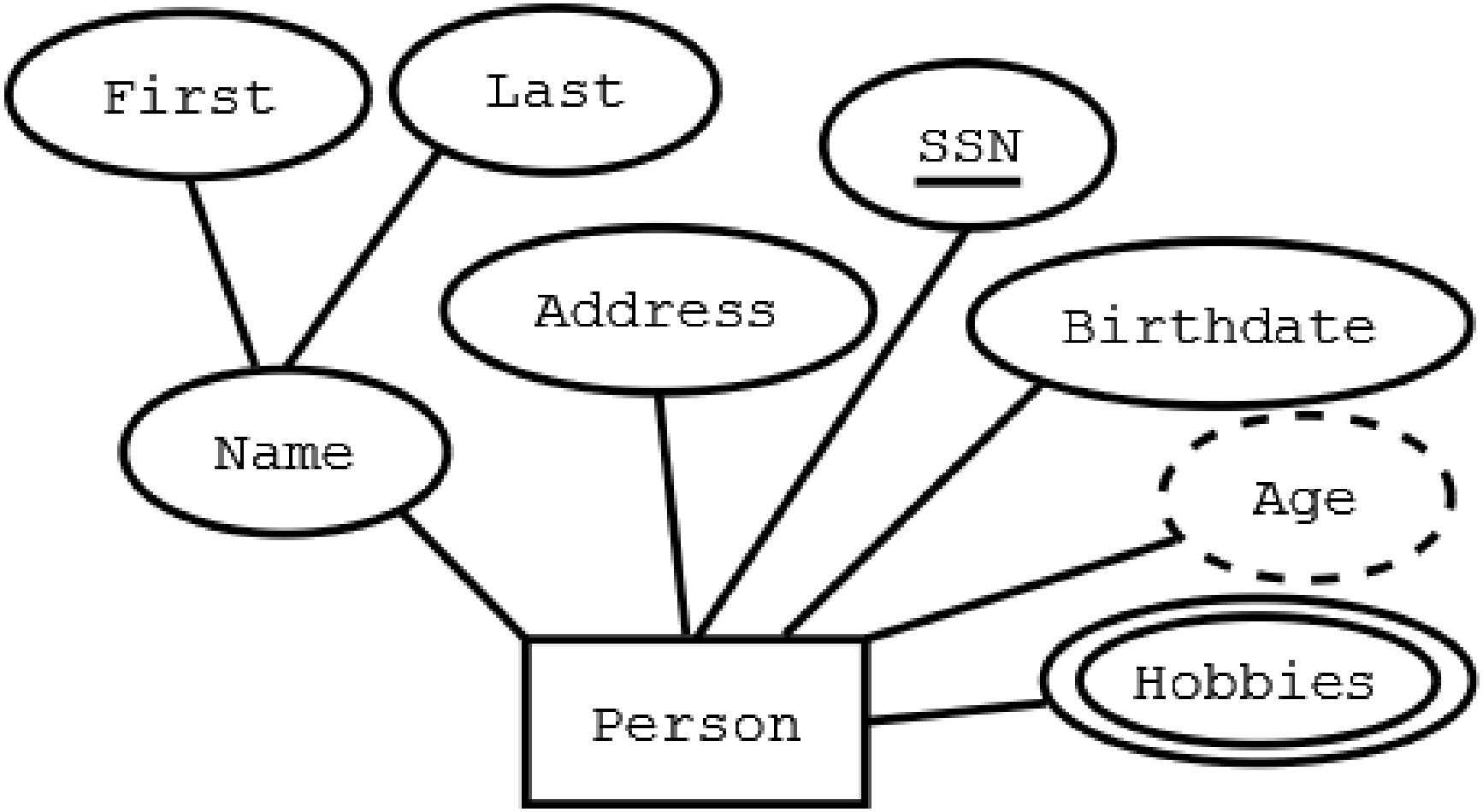
MULTIVALUED ATTRIBUTE

COMPOSITE ATTRIBUTE

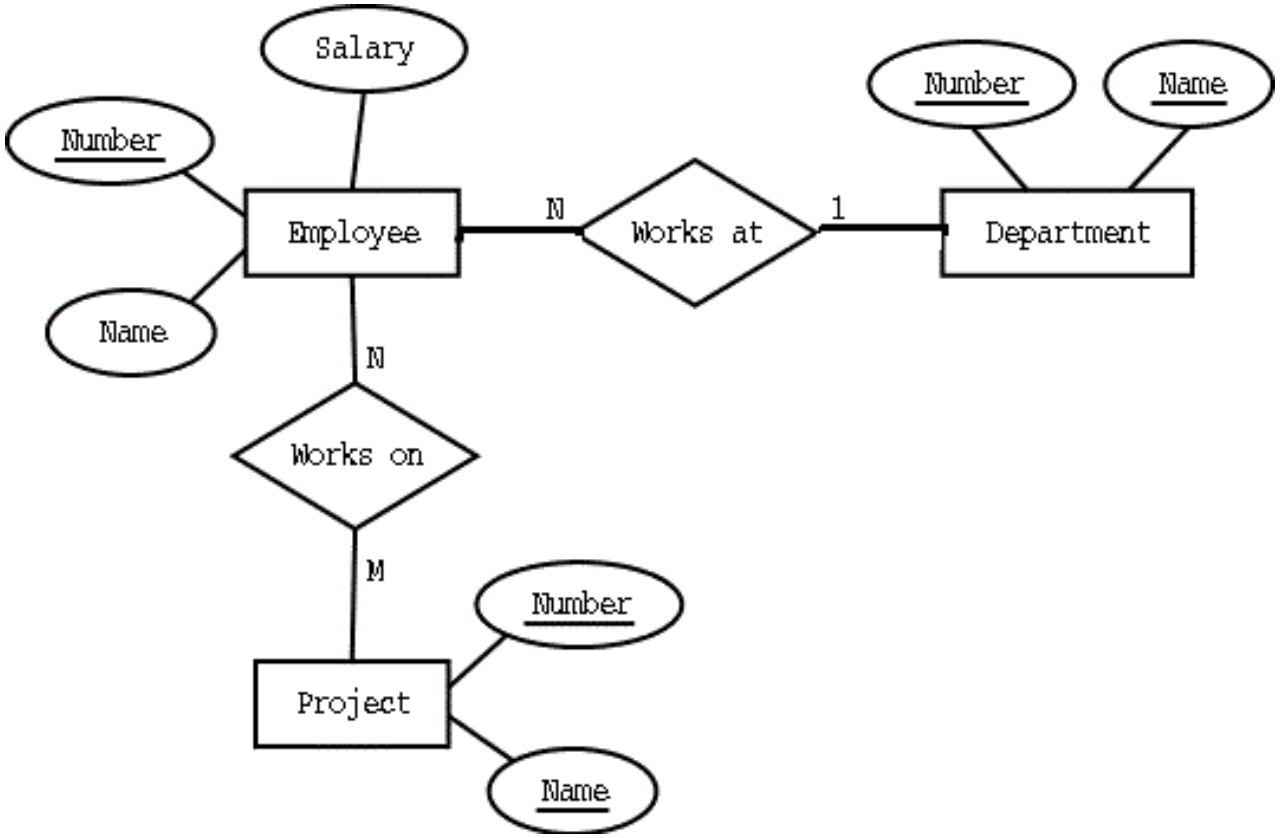
DERIVED ATTRIBUTE



# Attribute Notations



# The E-R Model - An Example



# The Different ER Relationships (cont..)

- Degree of Relationships
  - The degree of a relationship is the number of entities associated with the relationship

# Degree of a Relationship

- Unary:
- Recursive relationship between instances of ONE entity type.
  - Example: Salesperson MANAGES Salesperson.
- Binary:
  - Relationship between instances of TWO entity types (most common).
  - Example: Salesperson SELLS product.
- Ternary:
  - Relationship among instances of THREE entity types.
  - Example: Salesperson SELLS product TO customer.

# 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.

# 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.

# Goals of the Normalization

- Goals of the Normalization process are:
  - to eliminate redundant data
  - to ensure that data dependencies make sense
- Thus Normalization:
  - reduces the amount of space a database has consumed
  - ensures that data is logically stored

# 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

# Problems with un-normalized database

- An un-normalized database can suffer from various “logical inconsistencies” and “data operation anomalies”.
- Data Operation anomalies can be classified as:
  - Update anomaly
  - Insertion anomaly
  - Deletion anomaly

# Update anomaly

- The slide shows an Update anomaly.
- Employee 519 is shown as having different addresses on different records.

**Employees' Skills**

| Employee ID | Employee Address | Skill |
|---|---|---|
| 426 | 87 Sycamore Grove | Typing |
| 426 | 87 Sycamore Grove | Shorthand |
| 519 | 94 Chestnut Street | Public Speaking |
| 519 | 96 Walnut Avenue | Carpentry |

# Insertion anomaly

- The slide shows an example of an Insertion anomaly.
- Until the new faculty member is assigned to teach at least one course, the details cannot be recorded.

**Faculty and Their Courses**

| Faculty ID | Faculty Name | Faculty Hire Date | Course Code |
|---|---|---|---|
| 389 | Dr. Giddens | 10-Feb-1985 | ENG-206 |
| 407 | Dr. Saperstein | 19-Apr-1999 | CMP-101 |
| 407 | Dr. Saperstein | 19-Apr-1999 | CMP-201 |
| 424 | Dr. Newsome | 29-Mar-2007 | ? |

# Deletion anomaly

- The slide shows an example of a Deletion anomaly.
- All information about Dr. Giddens is lost when he temporarily ceases to be assigned to any course.

**Faculty and Their Courses**

| Faculty ID | Faculty Name | Faculty Hire Date | Course Code |
|---|---|---|---|
| 389 | Dr. Giddens | 10-Feb-1985 | ENG-206 |
| 407 | Dr. Saperstein | 19-Apr-1999 | CMP-101 |
| 407 | Dr. Saperstein | 19-Apr-1999 | CMP-201 |

DELETE

# Background to Normalization: Functional dependency

- Let us suppose we have two columns A and B, such that, for given value of column A there is a single value of column B associated with it. Then column B is said to be functionally dependent on column A.

- Column B is functionally dependent on column A is equivalent to saying that column A determines(identifies) column B.
  - The same can be notified as A ☐ B

- Eg: Employee Address has a functional dependency on Employee ID because a particular Employee ID value corresponds to one and only one Employee Address value.
  - EmpID ☐ EmpAddress

# Normal Forms

- Normal Form is abbreviated as NF.
- Normal Form provides criteria for determining a table's degree of vulnerability to logical inconsistencies and anomalies.
- The higher the NF applicable to a table, the less vulnerable it is to inconsistencies and anomalies.

# Un-normalized Relation

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

# 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.

# Normalization – First Normal Form (1 NF)

- A relation is said to be in "First Normal Form" (1NF) if and only if all its attributes assume only atomic values and there are no repeating groups.
  - 1NF requires that the values in each column of a table are "atomic".
  - "Atomic" implies that there are no sets of values within a column.
- To Transform un-normalized relation in 1 NF:
  - Eliminate repeating groups.
  - Make a separate table for each set of related attributes, and give each table a primary key.

# 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

**Table I**

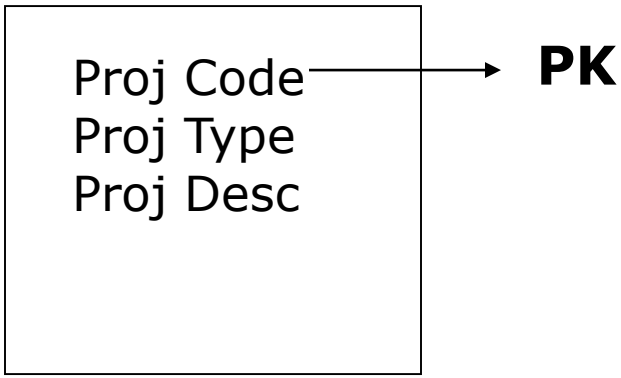
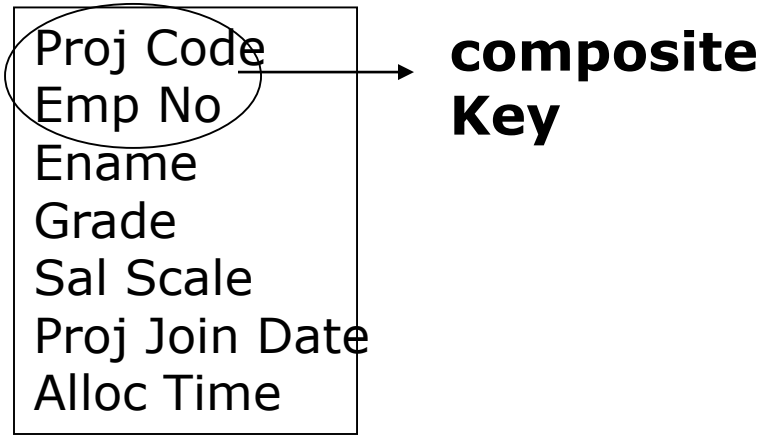



**Table II**



# Normalization – Second Normal Form (2 NF)

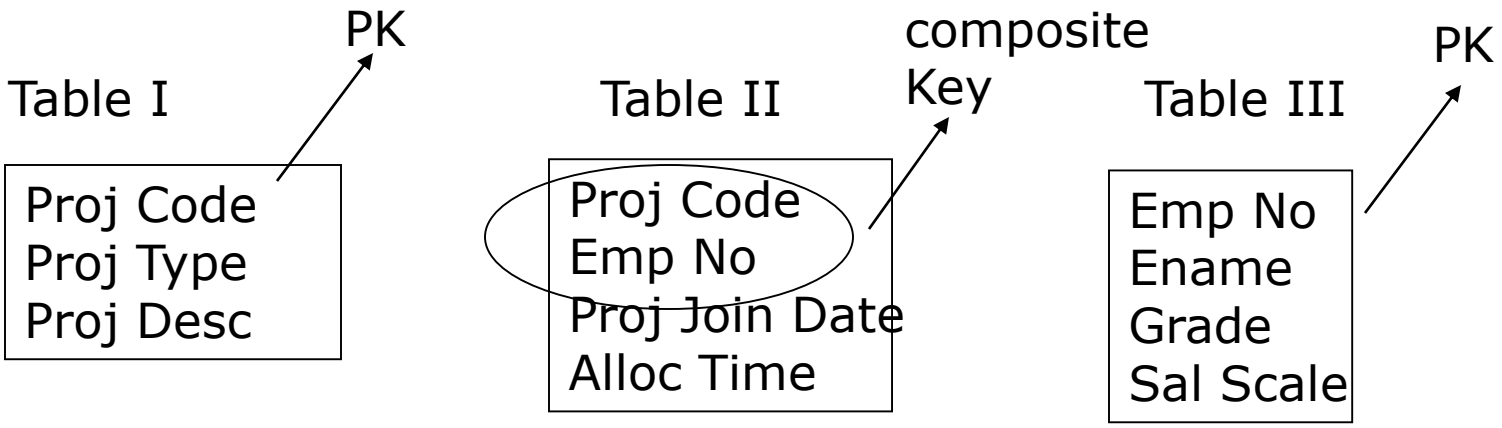
- A relation is in Second Normal Form (2NF) if and only if it is in 1NF and every non-key attribute is fully functionally dependent on the complete primary key of the relation.

- 2NF is based on the concept of Full Functional Dependency.

# Normalization – Second Normal Form (2 NF)

- Transform 1 NF to 2 NF :
  - If a non-key attribute depends on only part of a composite key,
  - remove it to a separate table.
  - For every relation with a single data item making up the primary key, this rule should “always be true”.
  - For those with the composite key, examine every column and ask whether its value depends on the whole of the composite key or just some part of it.
  - Remove those that depend only on part of the key to a new relation with that part as the primary key.

# Normalization – Second Normal Form (2 NF)



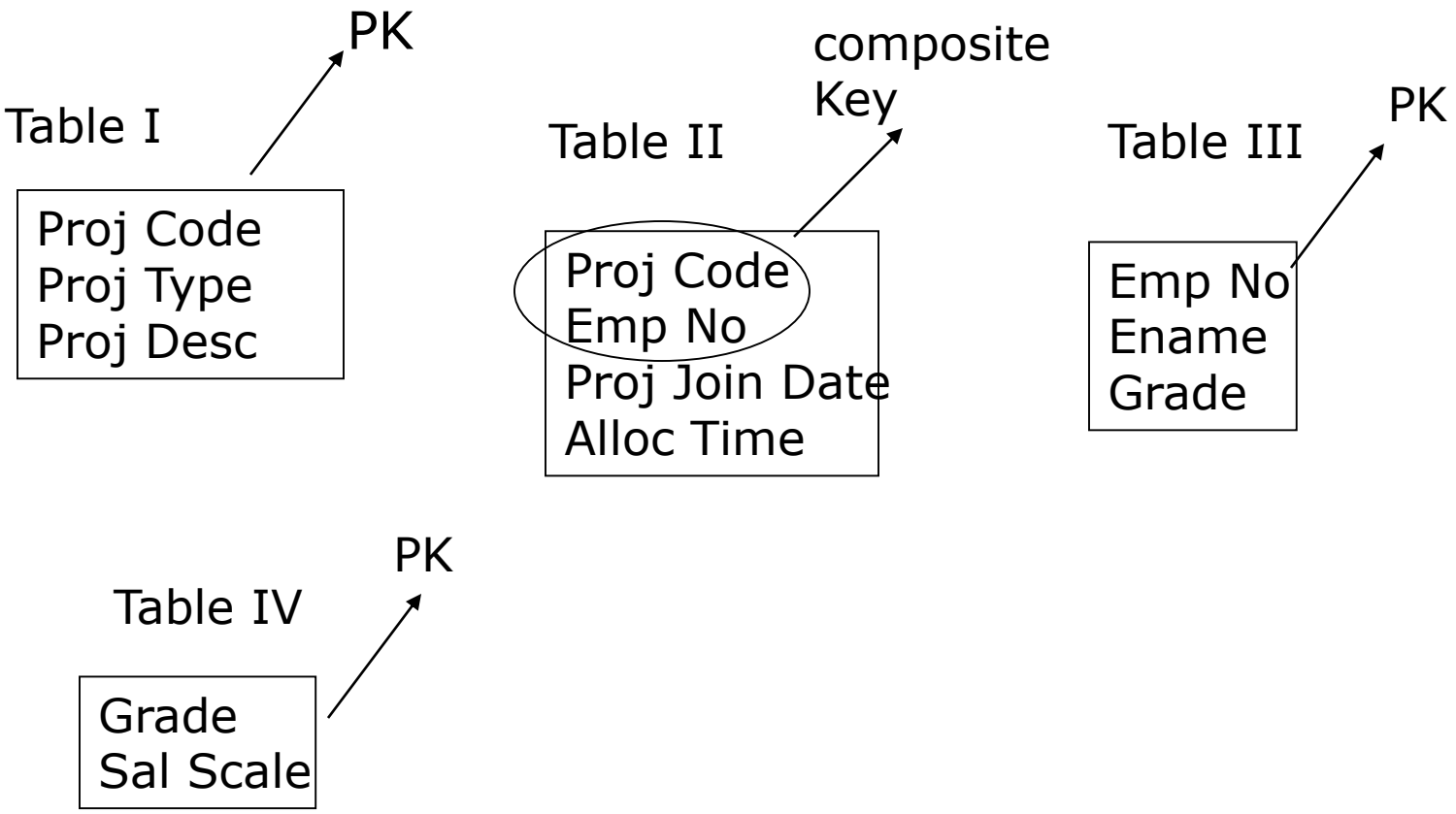
# Normalization – Third Normal Form (3 NF)

- A relation is in 3NF if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key of the relation.
- 3NF is based on the concept of transitive dependency.

# Normalization – Third Normal Form (3 NF)

- Rules for 3NF are:
  - Examine every non-key column and question its relationship with every other non-key column.
  - If there is a transitive dependency, then remove both the columns to a new relation.

# Normalization – Third Normal Form (3 NF)



# Normalization Summarization

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

# 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

# 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