

RDBMS - Bridge Course

Course Objectives

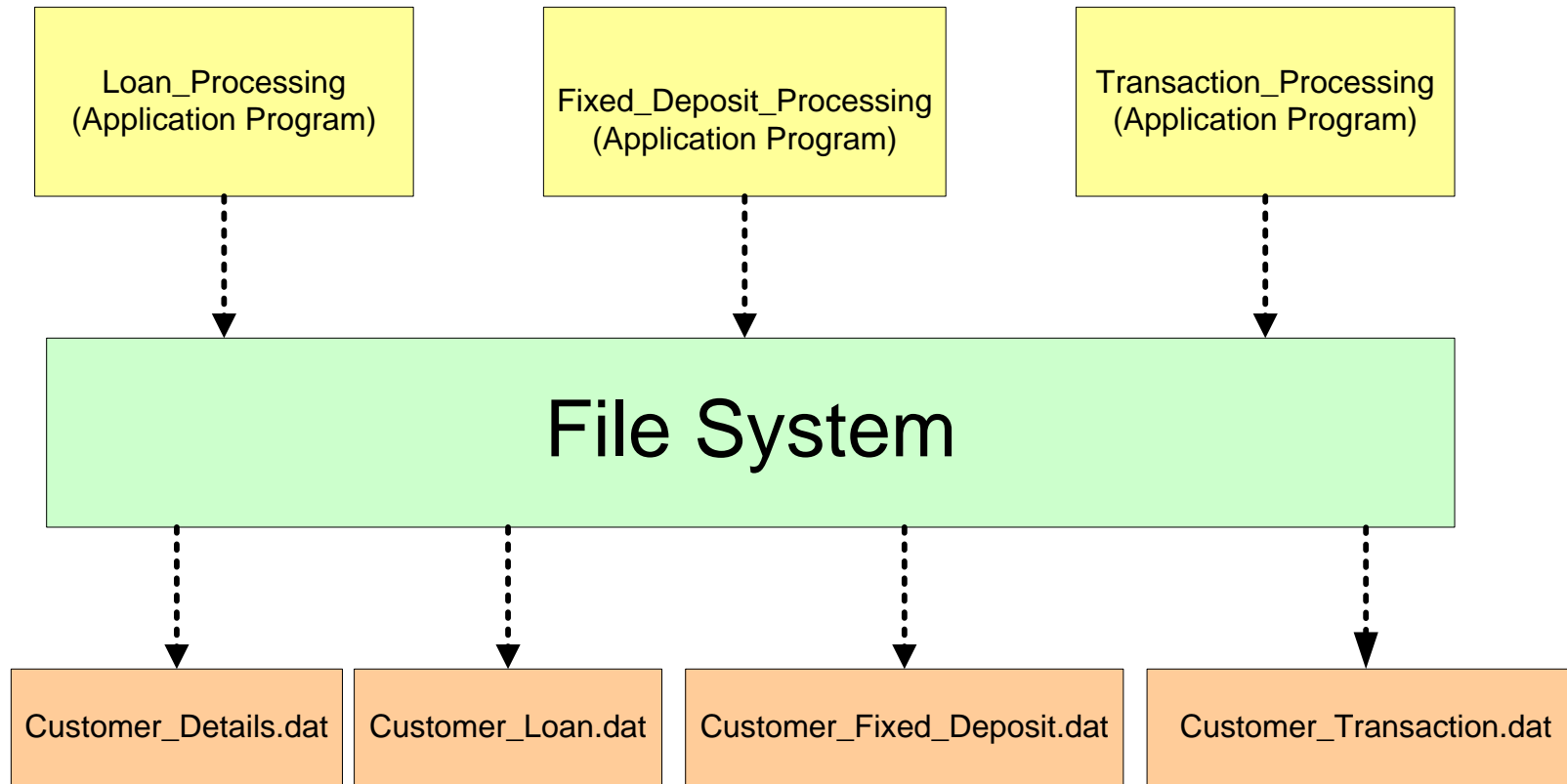
- To strengthen the basic RDBMS concepts.
- ER Modeling – Recap
- Normalization -Recap
- To strengthen the SQL concept
- To strengthen the concept of Joins and Sub query

Note: The course has been designed as a refresher course on RDBMS Concept.

Session Plan

- Traditional File Approach
- Advantages of a DBMS
- Relational Model Basics
- Keys
- Conceptual Design
 - ER Modelling
 - ER Modelling Notations
- Normalization

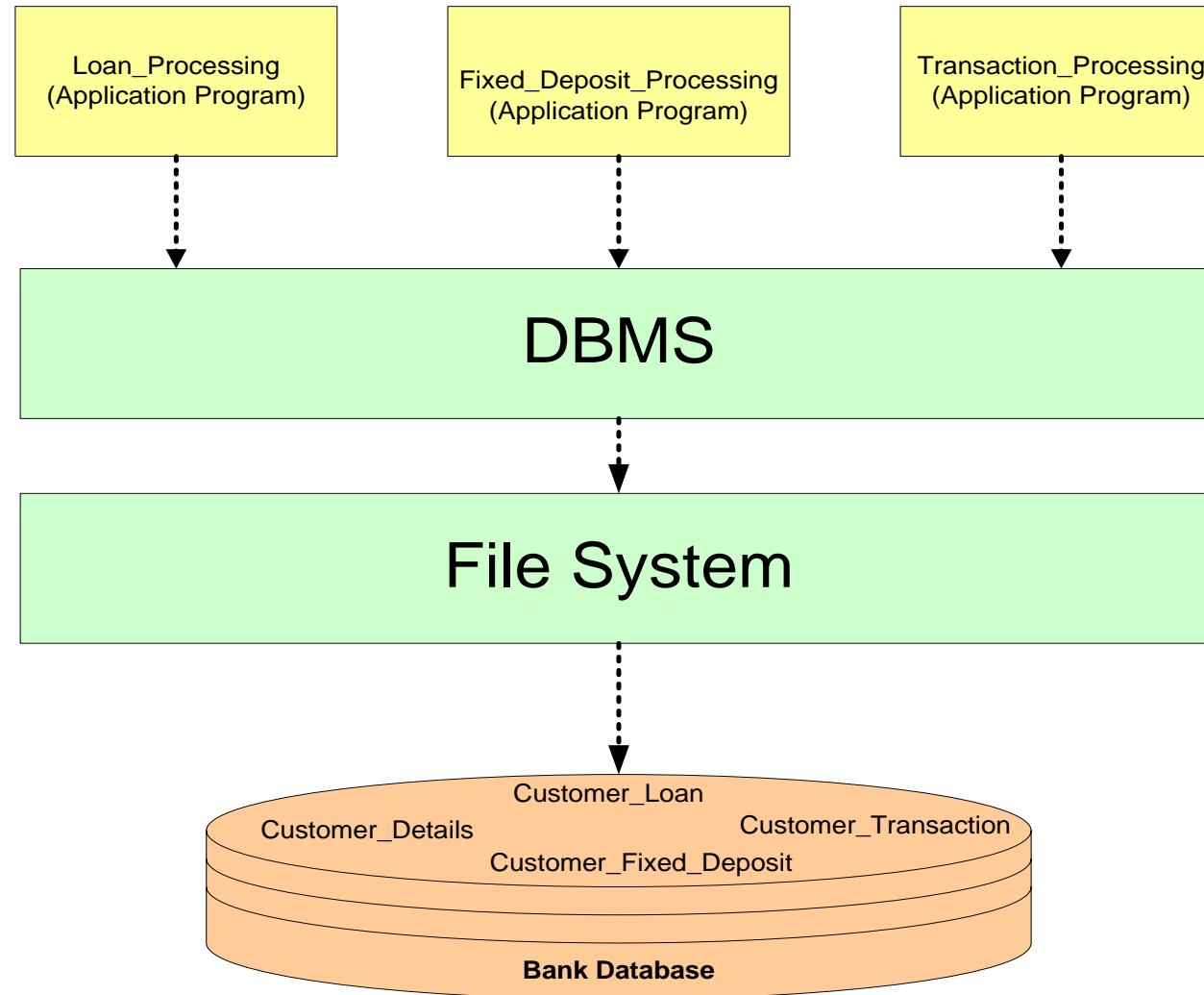
Traditional Method of Data Storage



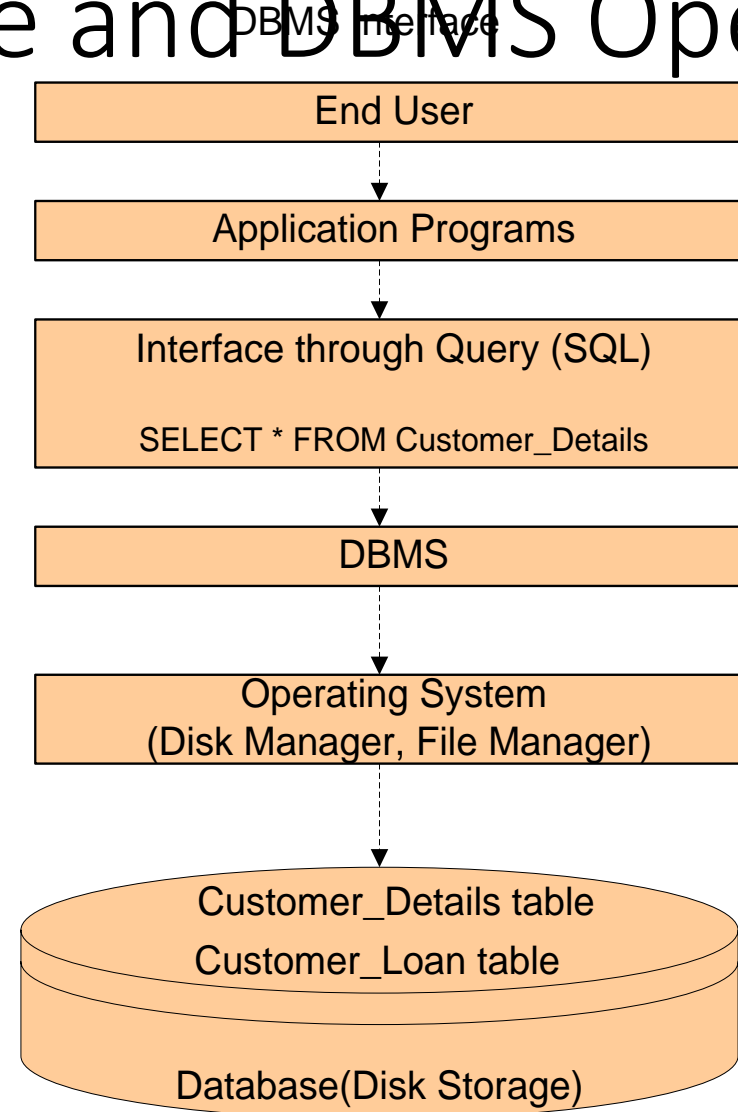
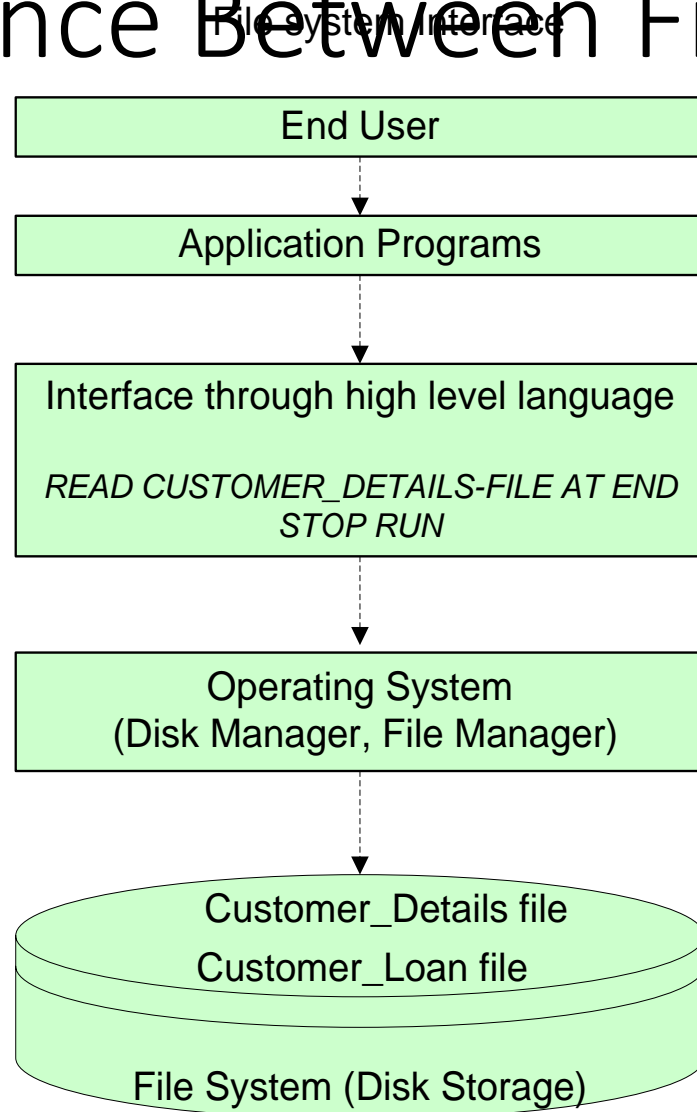
Database Management System

- Collection of interrelated files and set of programs which allows users to access and modify files
- Primary Goal is to provide a convenient and efficient way to store, retrieve and modify information
- Layer of abstraction between the application programs and the file system

Where does the DBMS fit in?



Difference Between File and DBMS Operations



Advantages of a DBMS

- Data independence
- Reduction in data redundancy
- Better security
- Better flexibility
- Effective data sharing
- Enforces integrity constraints
- Enables backup and recovery

Relational model basics

• Data is viewed as existing in two dimensional tables known as relations

- A relation (table) consists of unique attributes (columns) and tuples (rows)
- Sometimes the value to be inserted into a particular cell may be unknown, or it may have no value. This is represented by a **NULL**
- Null is not the same as zero, blank or an empty string
- Relational Database: Any database whose logical organization is based on relational data model.
- RDBMS: A DBMS that manages the relational database.

Keys in Relational Model

- Candidate key

A Candidate key is a set of **one or more attributes(minimal)** that can uniquely identify a row in a given table.

- Primary Key

During the creation of the table, the Database Designer chooses one of the Candidate Key from amongst the several available, to uniquely identify row in the given table.

- Alternate Key

The candidate key that is chosen to perform the identification task is called the *primary key* and the remaining candidate keys are known as alternate keys.

$$\text{No of Alternate Keys} = \text{No of Candidate Keys} - 1$$

Key and Non-key Attributes in Relational Model

- Key Attributes

The attributes that participate in the Candidate key are Key Attributes

- Non-Key Attributes

- The attributes other than the Candidate Key attributes in a table/relation are called Non-Key attributes.

OR

- The attributes which do not participate in the Candidate key.

Example

Given a relation

Trainee(Empno, FirstName, LastName, Email, PhoneNo)

Candidate key:

{Empno},{Email},{PhoneNo},{FirstName,LastName}

Primary key:

{Empno}

Alternate Key:

{Email},{PhoneNo},{FirstName,LastName}

Exercise on Key attributes

Given a relation $R1(X,Y,Z,L)$ and the following attribute(s) can uniquely identify the records of relation $R1$.

1)X

2)X,L

3)Z,L

Identify the following in relation $R1$?

Candidate Key(s)

Primary Key

Alternate Key

Key attribute(s)

Non-key attribute(s)

Foreign Key

- **Foreign key**

- A Foreign Key is a set of attribute (s) whose values are required to match values of a column in the same or another table.

DEPT

(Parent /Master/Referenced Table)

DeptNo	DName
D1	IVS
D2	ENR

EMP


(Child /Referencing Table)

EmpNo	EName	EDeptNo
1001	Elsa	D1
1002	John	D2
1003	Maria	Null
1004	Maida	D1

- **Point to remember**

- Foreign key values do not (usually) have to be unique.
- Foreign keys can also be *null* .

Foreign Key

- Foreign key 
 - A Foreign Key is a set of attributes of a table, whose values are required to match values of some Candidate Key in the same or another table
 - The constraint that values of a given Foreign Key must match the values of the corresponding Candidate Key is known as Referential constraint
 - A table which has a Foreign Key referring to its own Candidate Key is known as Self-Referencing table

Database Design Techniques

Top down Approach

Top down starts by defining the data sets and then define the data elements within those sets. As a result of this method, you generally end up with redundant information in one or more tables.

Some references call this **Entity - Relationship modeling**.

Bottom Up approach

Bottom up starts by defining the required attributes and then grouping them to form the entities. Another term used for this method is **normalization** from functional dependencies.

ER Modeling

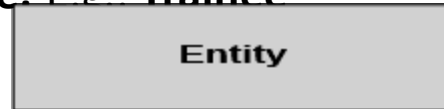
- Top down Approach

ER modeling

- **ER modeling:** A graphical technique for *understanding* and organizing the data independent of the actual database implementation.

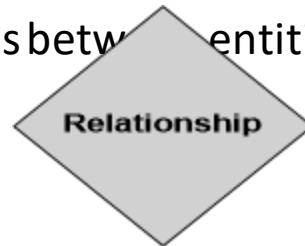
- **Entity:** Any thing that may have an independent existence and about which we intend to collect data.

Also known as **Entity type**. E.g.: **Trainee**



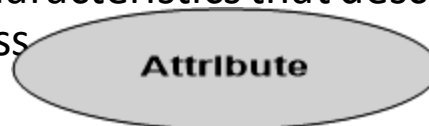
The Entity Type is represented as rectangle in ER Modeling

- **Relationships:** Associations between entities. E.g.: Trainee belongs to batch



The relationship type is represented as diamond in ER Modeling

- **Attributes:** Properties/characteristics that describe entities. eg: Trainee has BatchName, DOB, Address



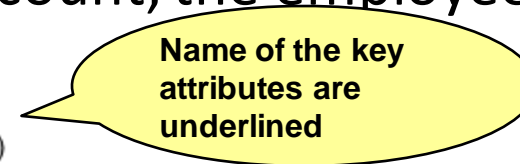
The attributes are represented as oval in ER Modeling

Attributes

- The set of possible values for an attribute is called the **domain** of the attribute

Example:

- The domain of attribute ***marital status*** is having four values: single, married, divorced or widowed.
- The domain of the attribute month is having twelve values ranging from January to December.
- **Key attribute:** The attribute (or combination of attributes) that is unique for every entity instance
 - E.g.: the account number of an account, the employee id of an employee etc.



Attribute Type

Types of Attributes	Definition	Example
Simple attribute	Cannot be divided into simpler components	Gender of the employee
Composite attribute	Can be split into components	Date of joining of the employee
Single valued	Can take on only a single value for each entity instance	Age of the employee
Multi-valued	Can take up many values	Skill set of the employee
Stored Attribute	Attribute that need to be stored permanently	Date of joining of the employee
Derived Attribute	Attribute that can be calculated based on other attributes.	Years of service of the employee

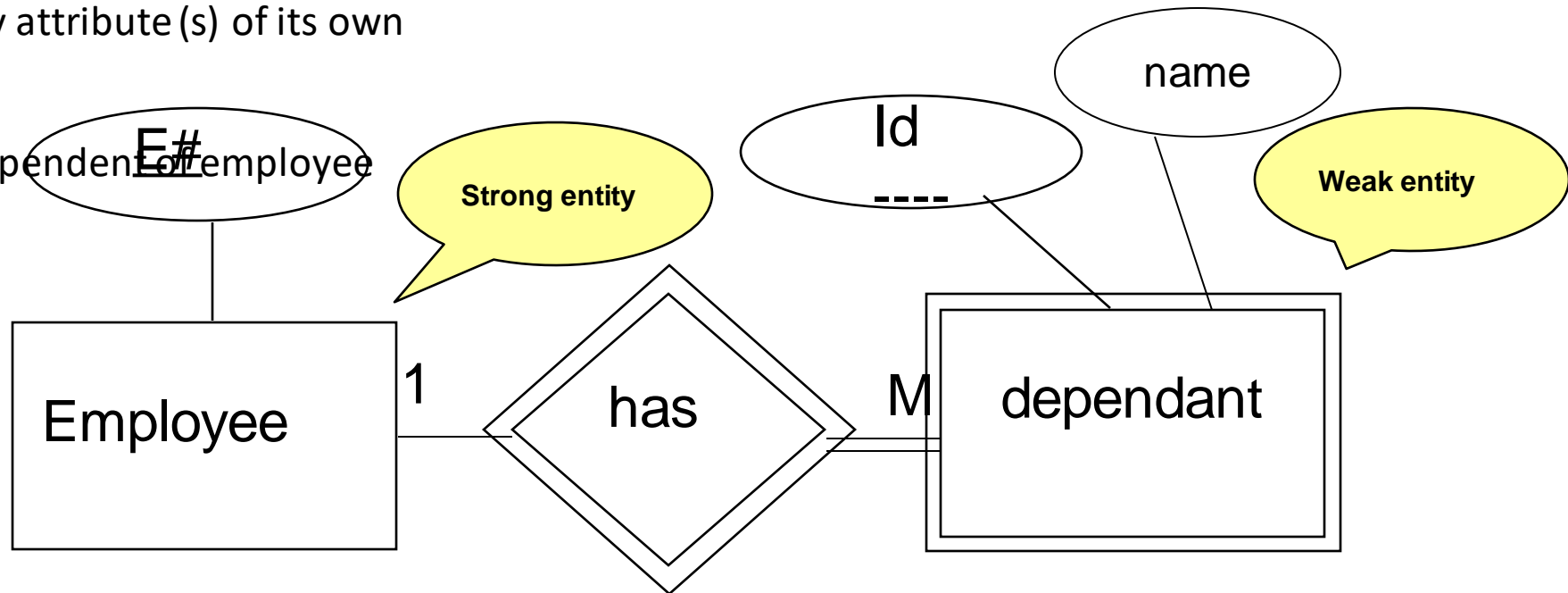
Entity Types

- **Regular Entity:** Entity that has its own key attribute (s).

E.g.: Employee, student, customer, policy holder etc.

- **Weak entity:** Entity that depends on other entity for its existence and doesn't have key attribute (s) of its own

E.g. : Dependent of employee



Degree of a Relationship

- **Degree:** the number of entity types involved
 - One *Unary*
 - Two *Binary*
 - Three *Ternary*

*E.g.: employee **manager-of** employee is unary*

*employee **works-for** department is binary*

*customer **purchase** item, shop keeper is a ternary relationship*

Cardinality

- Relationships can have different *connectivity*
 - **one-to-one** (1:1)
 - **one-to-many** (1:N)
 - **many-to-One** (M:1)
 - **many-to-many** (M:N)

E.g.:

Employee **head-of** department (1:1)

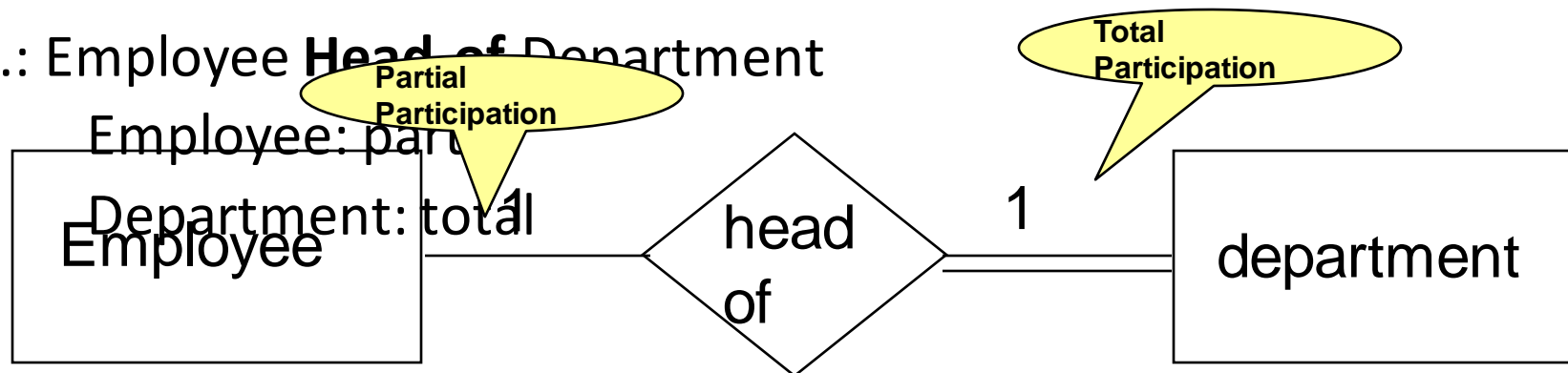
Lecturer **offers** course (1:N) assuming a course is taught by a single lecturer

Student **enrolls** course (M:N)

Relationship Participation

- **Total** : Every entity instance must be connected through the relationship to another instance of the other participating entity types
- **Partial**: All instances need not participate

E.g.: Employee Head of Department



ER Modeling -Notations



An Entity is an object or concept about which business user wants to store information.



A weak Entity is dependent on another Entity to exist. Example Order Item depends upon Order Number for its existence. Without Order Number it is impossible to identify Order Item uniquely.



Attributes are the properties or characteristics of an Entity



A key attribute is the unique, distinguishing characteristic of the Entity



A multi-valued attribute can have more than one value. For example, an employee Entity can have multiple skill values.

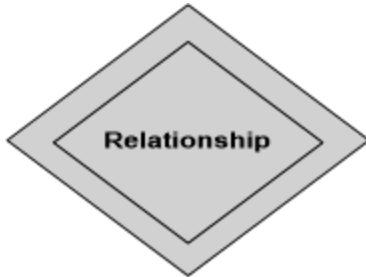
ER Modeling -Notations



A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's basic salary and House rent allowance.

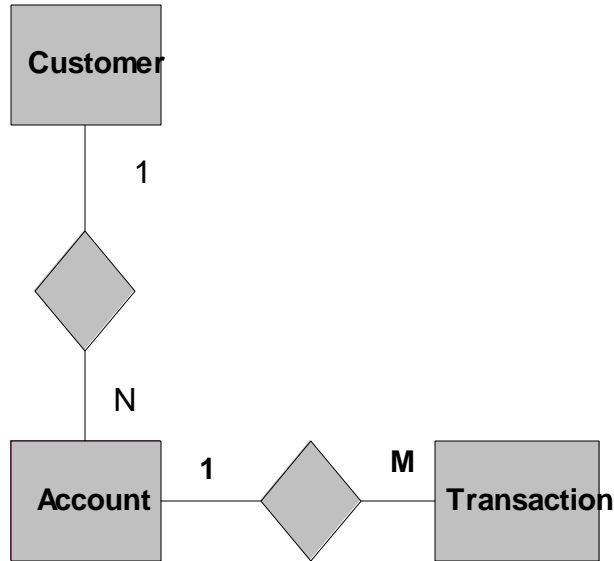


Relationships illustrate how two entities share information in the database structure.

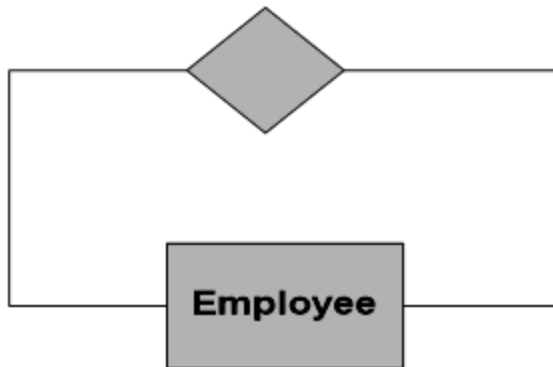


To connect a weak Entity with others, you should use a weak relationship notation.

ER Modeling -Notations

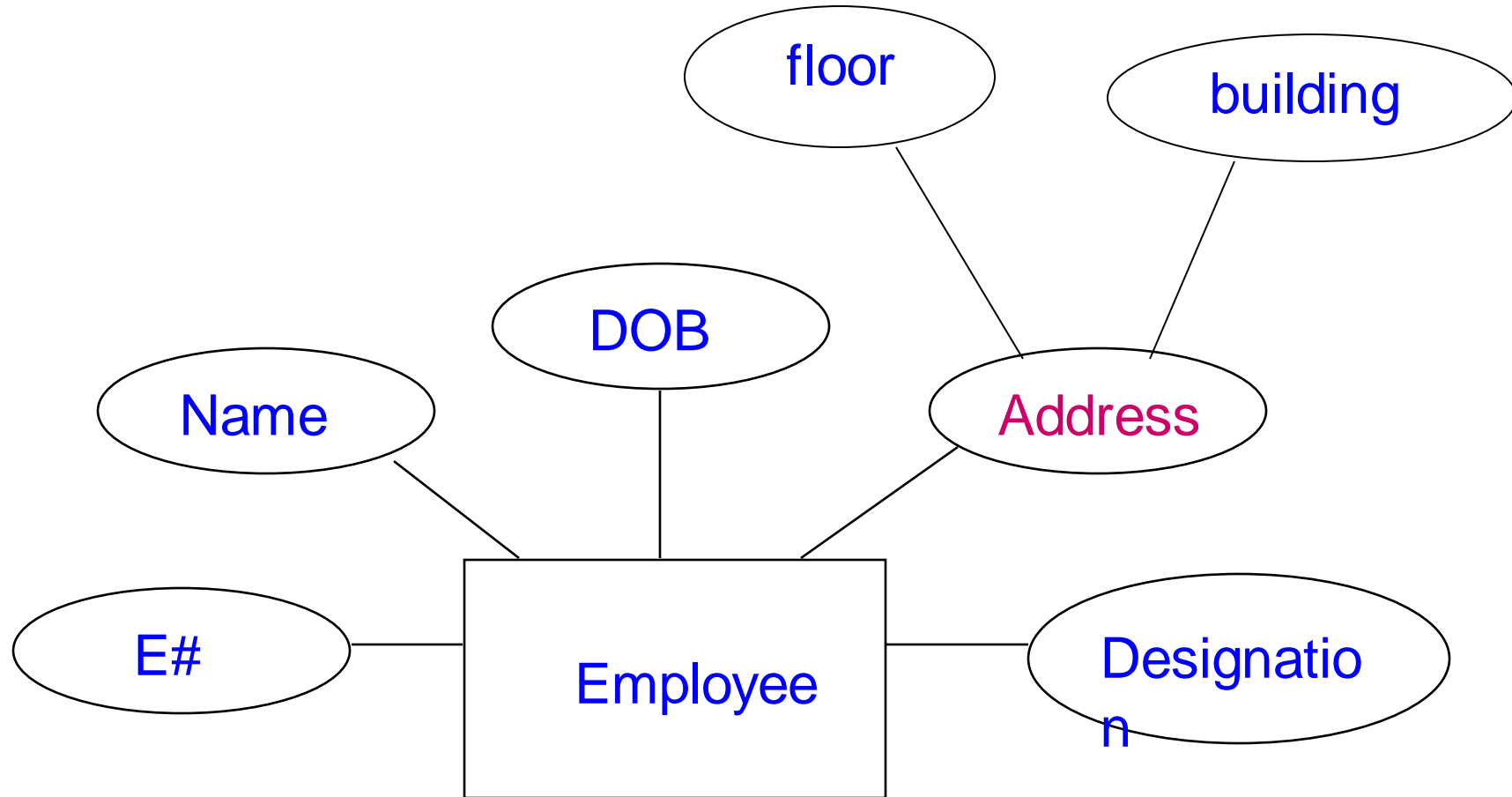


Cardinality specifies how many instances of an Entity relate to one instance of another Entity. M,N both represent 'MANY' and 1 represents 'ONE' Cardinality

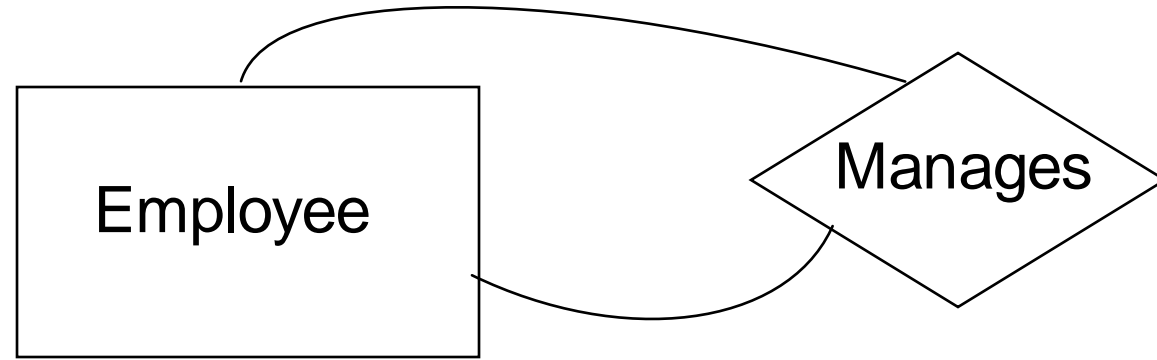


In some cases, entities can be self-linked. For example, employees can supervise other employees

Composite attribute

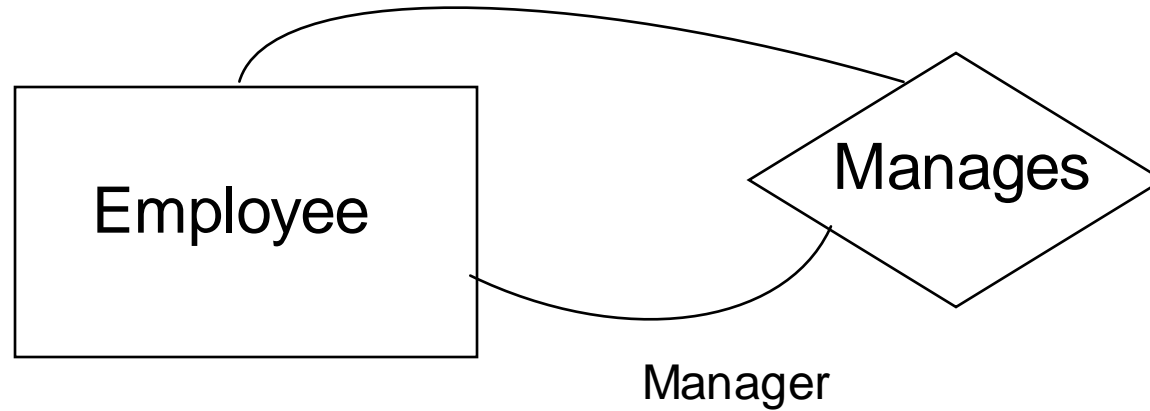


Unary Relationship

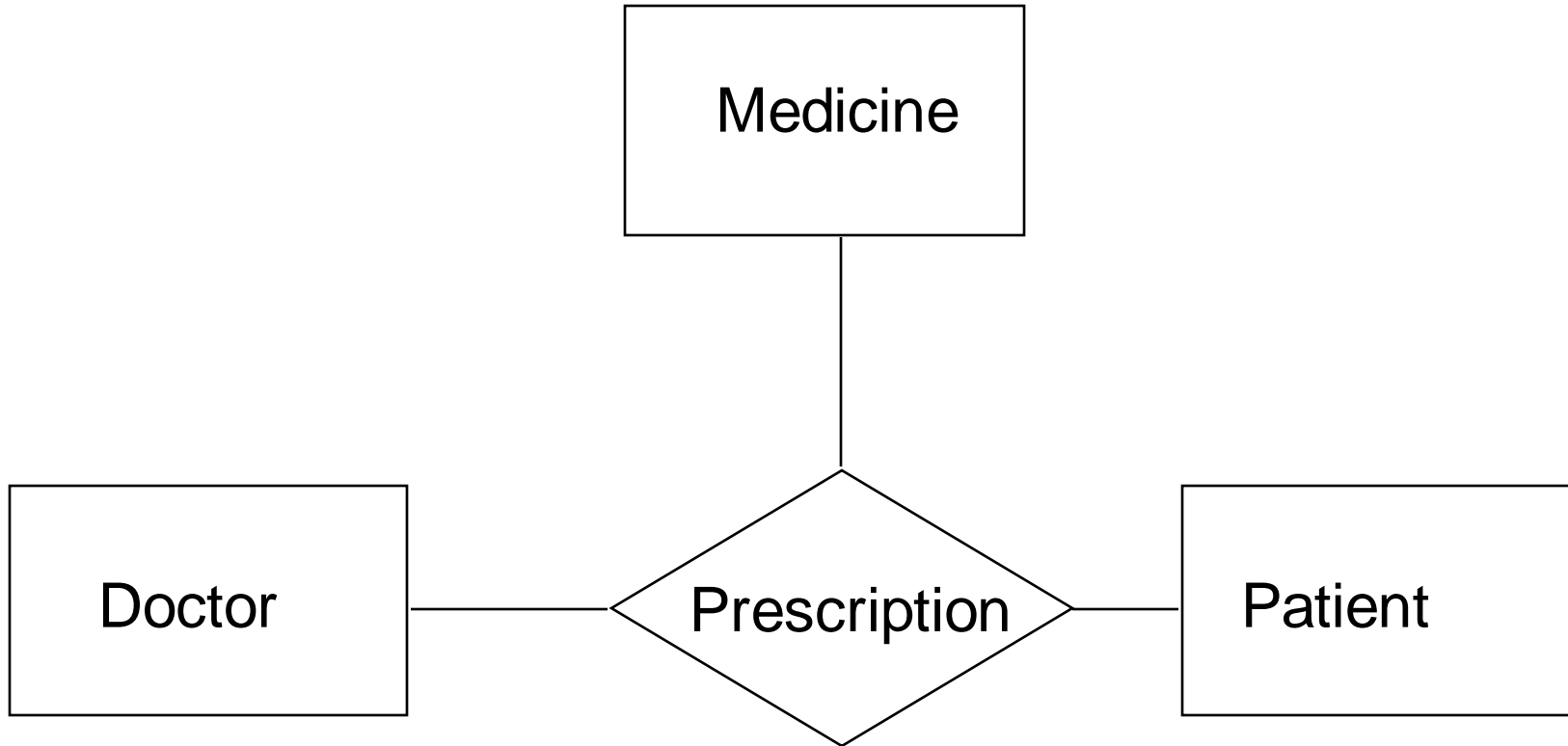


Role names

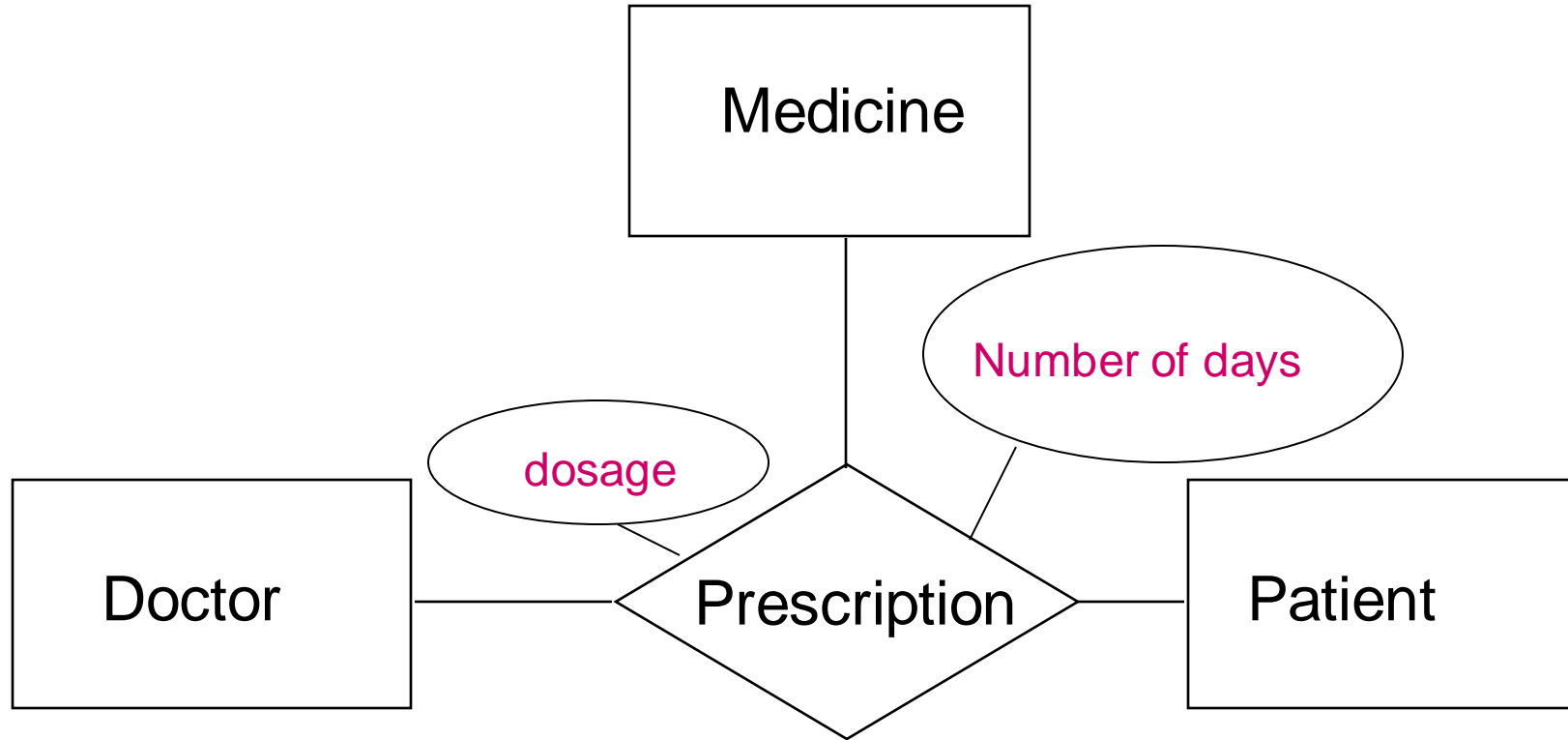
- Role names may be added to make the meaning more explicit



Ternary Relationship



Attributes of a Relationship



Steps in ER Modeling

- Identify the Entities
- Find relationships
- Identify the key attributes for every Entity
- Identify other relevant attributes
- Draw complete E-R diagram with all attributes including Primary Key
- Review your results with your Business users

ER Model For a college DB (Self Study)

Assumptions :

- A college contains many departments
- Each department can offer any number of courses
- Many instructors can work in a department
- An instructor can work only in one department
- For each department there is a Head
- An instructor can be head of only one department
- Each instructor can take any number of courses
- A course can be taken by only one instructor
- A student can enroll for any number of courses
- Each course can have any number of students

Step 1: Identify the Entities

- DEPARTMENT
- STUDENT
- COURSE
- INSTRUCTOR

Steps in ER Modeling (Self Study)

Step 2: Find the relationships

- One course is enrolled by multiple students and one student enrolls for multiple courses, hence the cardinality between course and student is Many to Many.
- The department offers many courses and each course belongs to only one department, hence the cardinality between department and course is One to Many.
- One department has multiple instructors and one instructor belongs to one and only one department, hence the cardinality between department and instructor is one to Many.
- Each department there is a “Head of department” and one instructor is “Head of department”, hence the cardinality is one to one.
- One course is taught by only one instructor, but the instructor teaches many courses, hence the cardinality between course and instructor is many to one.

Step 3: Identify the key attributes

- Deptname is the key attribute for the Entity “Department”, as it identifies the Department uniquely.
- Course# (CourseId) is the key attribute for “Course” Entity.
- Student# (Student Number) is the key attribute for “Student” Entity.
- Instructor Name is the key attribute for “Instructor” Entity.

Step 4: Identify other relevant attributes

- For the department entity, the relevant attribute is location
- For course entity, course name,duration,prerequisite
- For instructor entity, room#, telephone#
- For student entity, student name, date of birth

Step 5: Draw the E-R diagram



**ER diagram for the
University**

Normalization

- Bottom up approach

What is Normalization?

Database design may have some amount of

- Inconsistency
- Uncertainty
- Redundancy

To eliminate these drawbacks some **refinement** has to be done on the

database. This **Refinement** process is called **Normalization**.

Normalization

- It is Defined as a step-by-step process of decomposing a complex relation into a simple and stable data structure.
- It is a formal process that can be followed to achieve a good database design
- Also used to check that an existing design is of good quality
- The different stages of normalization are known as “**Normal Forms**”
- To accomplish normalization we need to understand the concept of **Functional Dependencies**.

Functional dependency

- In a given relation R, X and Y are attributes. Attribute Y is **functionally dependent** on attribute X if each value of X determines **EXACTLY ONE** value of Y, which is represented as $X \rightarrow Y$ (X can be composite in nature).
- We say here “x determines y” or “y is functionally dependent on x”
 $X \rightarrow Y$ does not imply $Y \rightarrow X$
- If the value of an attribute “Marks” is known then the value of an attribute “Grade” is determined since $\text{Marks} \rightarrow \text{Grade}$

Functional dependency

Types of functional dependencies:

- Full Functional dependency
- Partial Functional dependency
- Transitive dependency

Functional Dependencies

Consider the following Relation

REPORT (STUDENT#,COURSE#, StudentName, CourseName, Marks, Grade)

Description of the Attributes:

- STUDENT# - Student Number
- COURSE# - Course Number
- StudentName- Student Name
- CourseName - Course Name
- Marks - Scored in Course COURSE# by Student STUDENT#
- Grade - obtained by Student STUDENT# in Course COURSE#

Functional Dependencies- From the previous example

- For each value of (Student# ,Course#), Marks obtained will be exactly one value. So we observe the following Functional dependency

STUDENT# COURSE# → Marks

- For each value of Course# the name of the course will be exactly one value. So we observe the following Functional dependency

COURSE# → CourseName,

- For each value of Marks the grade will be exactly one value. So we observe the following functional dependency

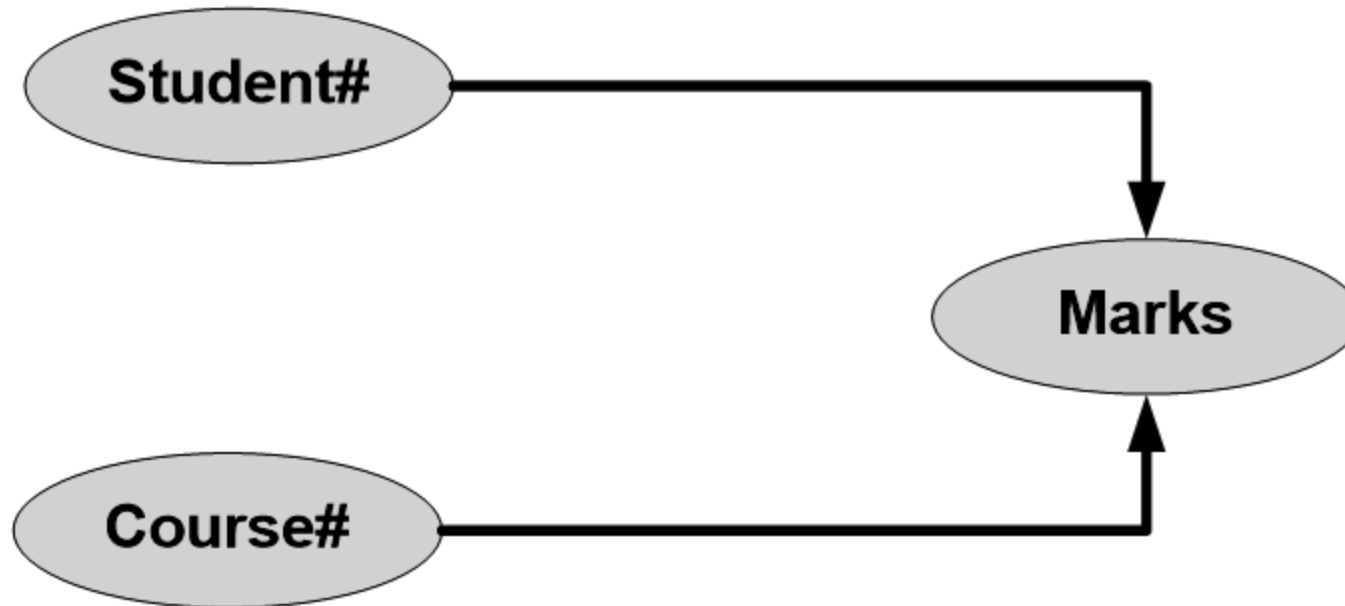
Marks → Grade

Full dependencies

X and Y are attributes.

X Functionally determines Y

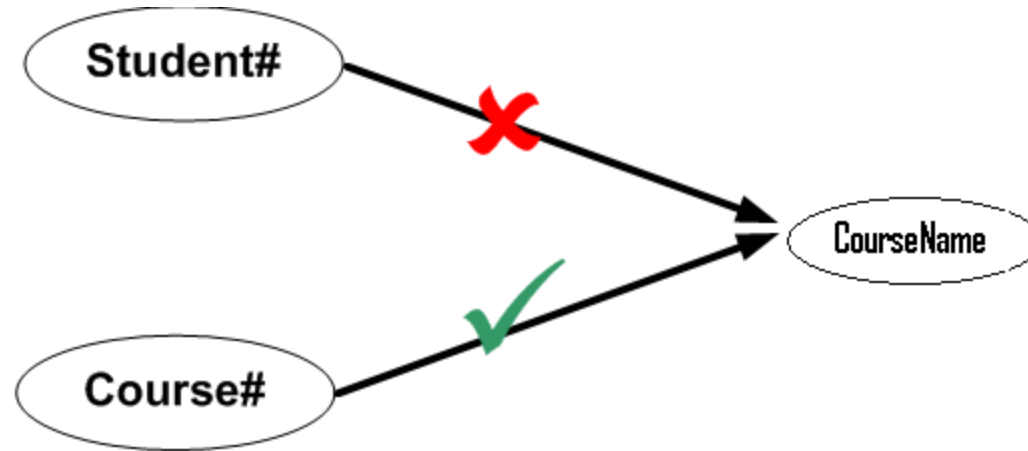
Note: Subset of X should not functionally determine Y



Partial dependencies

X and Y are attributes.

Attribute Y is partially dependent on the attribute X only if it is dependent on a sub-set of attribute X.



We have both the functional dependency valid in our example

Student# Course# \longrightarrow CourseName

Course# \longrightarrow CourseName

So we can say that CourseName is partially dependent on Student# Course#

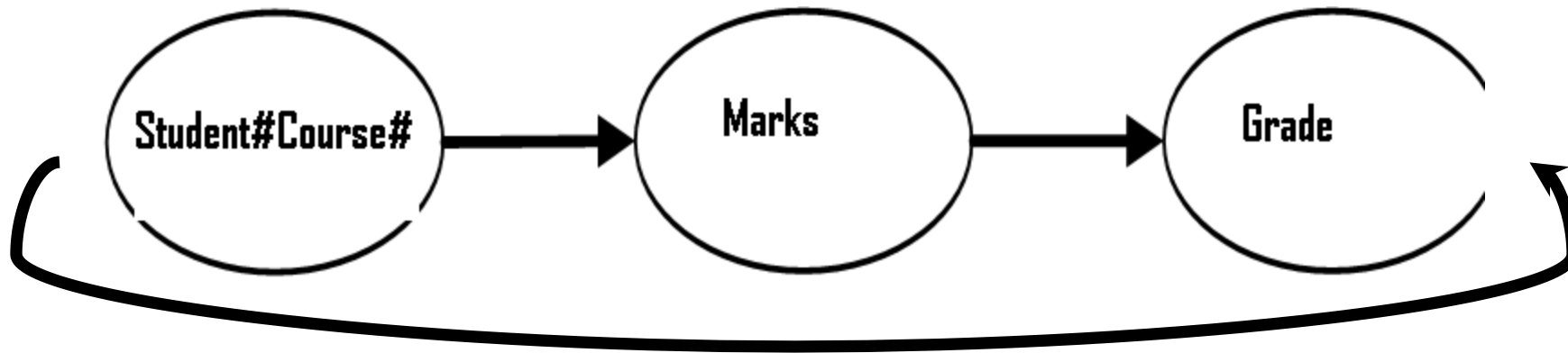
Transitive dependencies

X Y and Z are three attributes.

$X \rightarrow Y$

$Y \rightarrow Z$

$\Rightarrow X \rightarrow Z$



Need for Normalization

Lets observe the Online Retail Application Table

CustomerDetails	ItemDetails	PurchaseDetails
1001 John 1500012351	STN001 Pen 10 A	5 50
1002 Tom 1200354611	BAK003 Bread 10 A	1 10
1003 Maria 2134724532	GRO001 Potato 20 B	1 20

Each row of the table Represents a customer who has purchased an item.

Need for Normalization

In this Scenario

➤ Can we Insert the record of an item which has not been purchased by any customer?

The table is not to maintain the record of items but it is to keep the record of purchase of item by customers

➤ Can we delete the record of item which was purchased by one customer?

➤ How many rows we need to update if there is a change in description of item?

There will be information Loss for that item

➤ How many times we need to store the description of an item if the same item is purchased many times?

Depends upon the no of times the item has been purchased

So we observe the following in the Un Normalized table:

**Insert , Delete, Update Anomaly
Data Duplication**

Depends upon the no of times the item has been purchased

Need for Normalization

So we observe the following in the Un Normalized table:

- Insert , Delete, Update Anomaly
- Data Duplication

First Normal Form: 1NF

- **A relation schema is in 1NF :**
 - if and only if all the attributes of the relation R are atomic in nature.
 - **Atomic:** the smallest level to which data may be broken down and remain meaningful

Online Retail Application Tables – 1NF Normalized

Observation on Un Normalized Retail Application Table

CustomerDetails	ItemDetails	PurchaseDetails
1001 John 1500012351	STN001 Pen 10 A	5 50
1002 Tom 1200354611	BAK003 Bread 10 A	1 10
1003 Maria 2134724532	GRO001 Potato 20 B	1 20

Above observation violates 1NF definition
To bring it to 1NF we need to make the columns atomic

CustomerId	CustomerName	Accountno	ItemId	ItemName	UnitPrice	Class	QtyPurch ased	NetAmt
1001	John	1500012351	STN001	Pen	10	A	5	50
1002	Tom	1200354611	BAK003	Bread	10	A	1	10
1003	Maria	2134724532	GRO001	Potato	20	B	1	20

Second Normal Form: 2NF

- A Relation is said to be in Second Normal Form if and only if :
 - It is in the First normal form, and
 - No partial dependency exists between non-key attributes and key attributes.

- An attribute of a relation R that belongs to the candidate key of R is said to be a **key attribute** and that which doesn't is a **non-key attribute**

To make a table 2NF compliant, we have to remove all the partial dependencies

Second Normal Form : Example

Functional Dependencies of Retail Application Table

RetailApplicationTable(CustomerId, ItemId, CustomerName, AccountNo, ItemName, UnitPrice, Class, QtyPurchased, NetAmount)

- | | | |
|----------------------|---|----------------------------|
| (i) CustomerId | → | CustName, AccountNo |
| (ii) ItemId | → | ItemName, UnitPrice, Class |
| (iii) CustId, ItemId | → | QtyPurchased, NetPrice |
| (iv) UnitPrice | → | Class |

Second Normal Form : Example (Cont..)

Key and Non Key Attributes of Retail Application Table

{CustomerId, ItemId} is Candidate key

Key Attributes: CustomerId, ItemId

Non Key Attributes: CustomerName,
AccountNo,
ItemName,
UnitPrice,
Class,
QtyPurchased,
NetAmount

Second Normal Form : Example (Cont..)

Fully Functionally dependent on Key Attribute

CustomerId, ItemId \longrightarrow QtyPurchased, NetPrice

Partial Dependency with respect to Key Attribute

CustomerId \longrightarrow CustomerName, AccountNo

ItemId \longrightarrow ItemName, UnitPrice, Class

No Dependency with respect to Key Attribute

UnitPrice \longrightarrow Class

Second Normal Form : Example (Cont..)

After removing the Partial dependencies on Key Attributes we get the below tables which are in 2NF:

Customer

CustomerId	CustomerName	Accountno
1001	John	1500012351
1002	Tom	1200354611
1003	Maria	2134724532

Item

ItemId	ItemName	UnitPrice	Class
STN001	Pen	10	A
BAK003	Bread	10	A
GRO001	Potato	20	B

Second Normal Form : Example (Cont..)

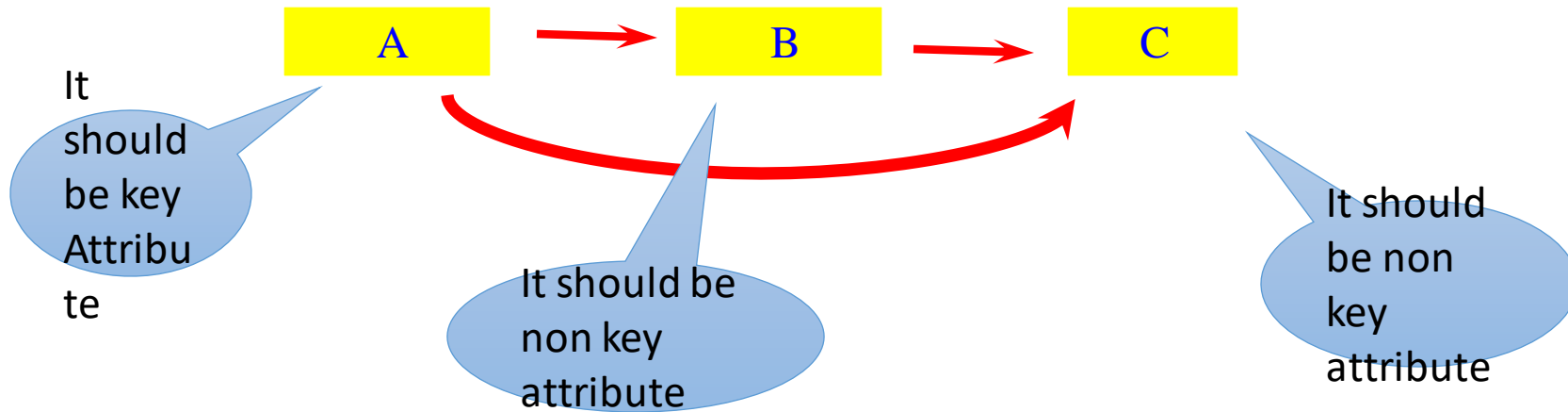
ItemPurchase

CustomerId	ItemId	QtyPurchased	NetAmt
1001	STN001	5	50
1002	BAK003	1	10
1003	GRO001	1	20

Third Normal Form: 3 NF

A relation R is said to be in the Third Normal Form (3NF) if and only if

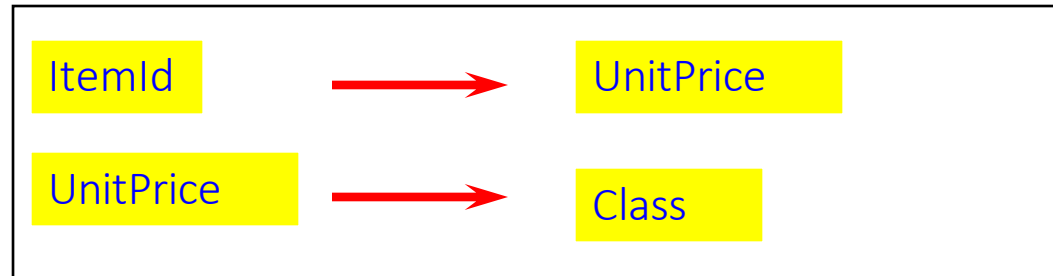
- It is in 2NF and***
- No transitive dependency exists between non-key attributes and key attributes through another non key attribute.***



To make a table 3NF compliant, we have to remove all such Transitive Dependencies

Third Normal Form : Example

Let us consider Item table obtained after bringing Retail Application table in to 2NF:



The above dependencies violates 3NF definition

Third Normal Form : Example (Cont.)

After removing the transitive dependencies from the Item table we get the following two table

Item

ItemId	ItemName	UnitPrice
STN001	Pen	10
BAK003	Bread	10
GRO001	Potato	20

ItemClass

UnitPrice	Class
10	A
20	B

Merits of Normalization

- Normalization is based on a mathematical foundation.
- Removes the redundancy to a greater extent.
- Removes the anomalies present in INSERTs, UPDATEs and DELETEs.

Demerits of Normalization

- Data retrieval or SELECT operation performance will be severely affected.
- Normalization might not always represent real world scenarios.

Summary of Normal Forms

Input	Operation	Output
Un-normalized Table	Create separate rows or columns for every combination of multi valued columns	Table in 1 NF
Table in 1 NF	Eliminate Partial dependencies	Tables in 2NF
Tables in 2 NF	Eliminate partial dependency	Tables in 3 NF

Summary – ER Model

- Most of the application errors are because of miscommunication between the application user and the designer and between the designer and the developer.
- It is always better to represent business findings in terms of picture to avoid miscommunication
- It is practically impossible to review the complete requirement document by business users.
- An E-R diagram is one of the many ways to represent business findings in pictorial format.
- E-R Modeling will also help the database design
- E-R modeling has some amount of inconsistency and anomalies associated with it.

Summary - Normalization

- Normalization is a refinement process. It helps in removing anomalies present in INSERTs/UPDATEs/DELETEs.
- Normalization is also called “**Bottom-up approach**”, because this technique requires very minute details like every participating attribute and how it is dependant on the key attributes, is crucial. If you add new attributes after normalization, it may change the normal form itself.
- There are three normal forms that were defined being commonly used.
- 1NF makes sure that all the attributes are atomic in nature.
- 2NF removes the partial dependency.

Summary

- Traditional File Approach
- Advantages of a DBMS
- Relational Model Basics
- Keys
- Conceptual Design
 - ER Modelling
 - ER Modelling Notations
- Normalization

Thank You