Relational Database Management System

Agenda

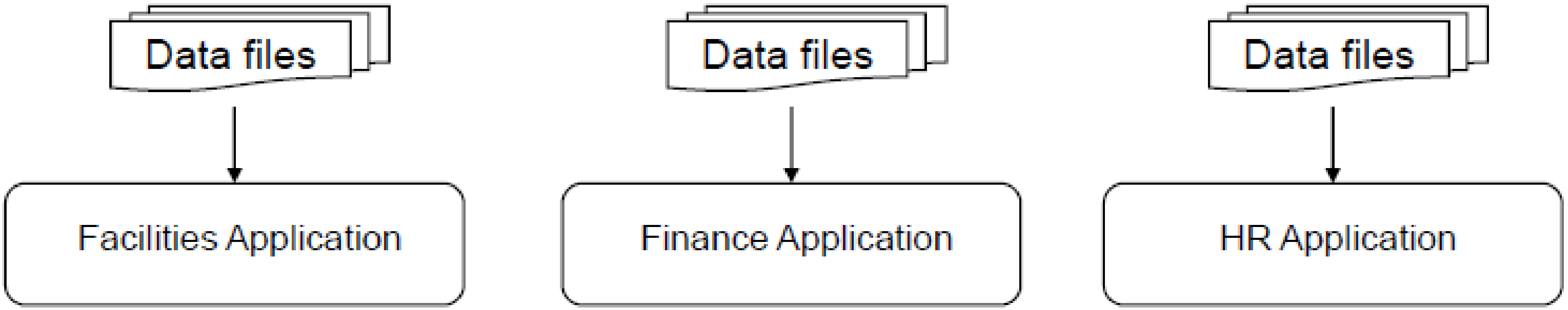
- RDBMS Concepts
- ER Modeling
- Database design
- SQL Lab

Data versus Information

- What is data?
  - Plural of datum
  - Represents facts concerning people, objects, events, etc.,
- Examples:
  - Sanjeev 220456
  - Rashmi 242056
- What is information?
  - Data that has been processed and presented in a meaningful format
  - Increases our sense of awareness about facts
- Examples:
  - Sanjeev's annual CTC is 220456
  - Rashmi's PeopleSoft ID is 242056

Traditional methods of Information management

- Individual applications were developed to meet specific user requirements
- Dedicated data files were created for each application



Techniques used with data files

4176	Aniruddha Sarkar	CAPS
4181	Manoj Saha	WENA
4183	Moushumi Dharchoudhury	CENA
4203	Suryanarayana D.V.S.S.	SONA
4204	Vivek Rai	CAPS

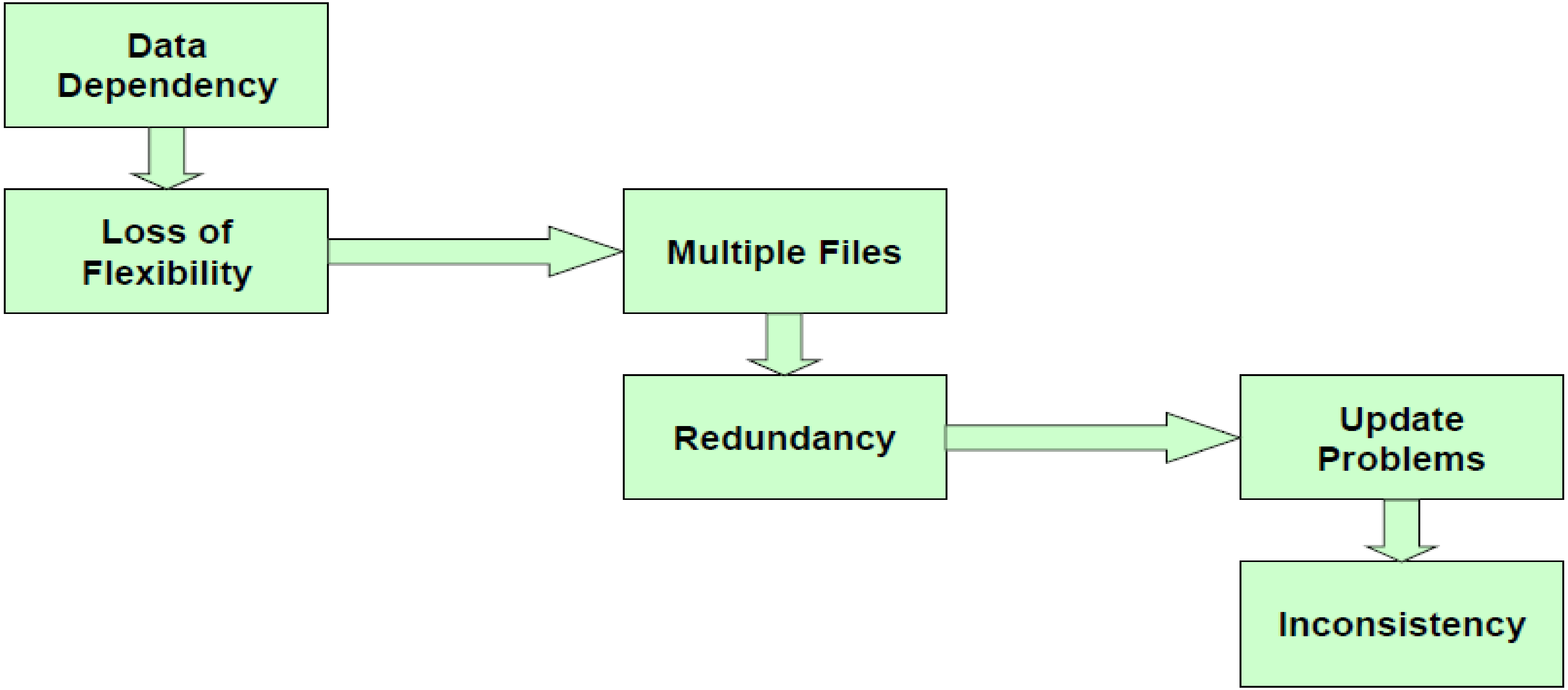
Predefined length

```
4176  AniruddhaSarkar  CAPS
4181  ManojSaha  WENA
4183  MoushumiDharchoudhury  CENA
4203  SuryanarayanaD.V.S.  SONA
4204  VivekRai       CAPS
```

Use space / comma or some other special character as a separator

4176,Aniruddha Sarkar,CAPS,4181,Manoj Saha,WENA,4183,Moushumi Dharchoudhury,CENA,4203,Suryanarayana D.V.S.,SONA,4204,Vivek Rai,CAPS

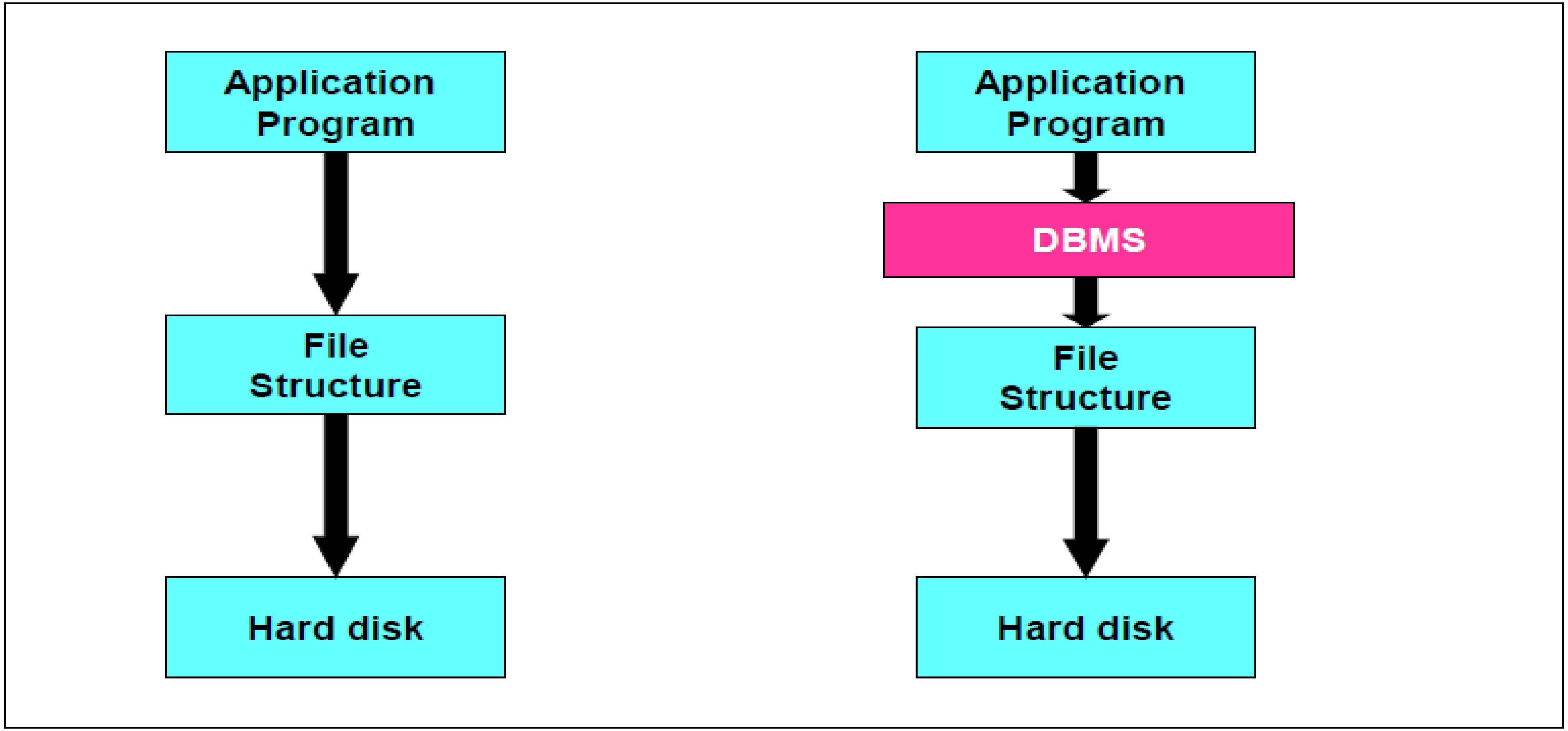
Problems with File based Systems



Database & DBMS

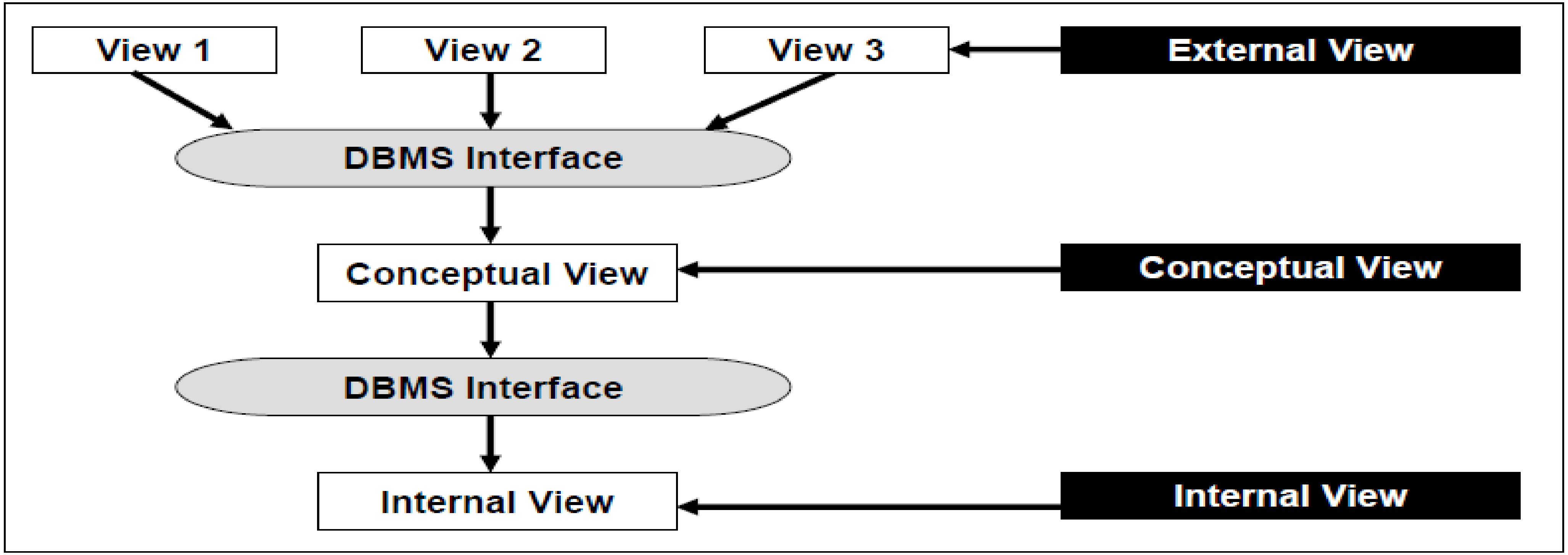
- Database
  - Is a shared repository of inter-related data
  - Represents objects in the real world and relationship between them
  - At the physical level a database is just a set of files
- Database Management System (DBMS)
  - Is a software used to create, manage & monitor databases
  - Is an interface between the user and the database

Where does the DBMS fit?



Three-tier architecture of a DBMS

- Most of the commercial DBMS are based on a 3-tier architecture model called ANSI/SPARC (American National Standards Institute, Standard Planning and Requirements Committee) as shown below.



Three-tier architecture of a DBMS - Example

External View

```
EMPVIEW1
        EMPNO           CHARACTER (4),
        EMPNAME         VARCHAR2(15),
```

Conceptual View

```
EMPLOYEE (
        EMPNO           CHARACTER (4),
        EMPNAME         VARCHAR2(15),
        DEPT            CHARACTER (4)
        SALARY          DECIMAL(8,2));
```

Internal View

```
EMPLOYEE LENGTH 31
   EMPNO      CHARACTER (4)    OFFSET = 0,    INDEX = EMPX
   EMPNAME    VARCHAR2(15)     OFFSET = 4,
   DEPT       CHARACTER (4)    OFFSET = 19,
   SALARY     DECIMAL(8,2)     OFFSET = 23
```

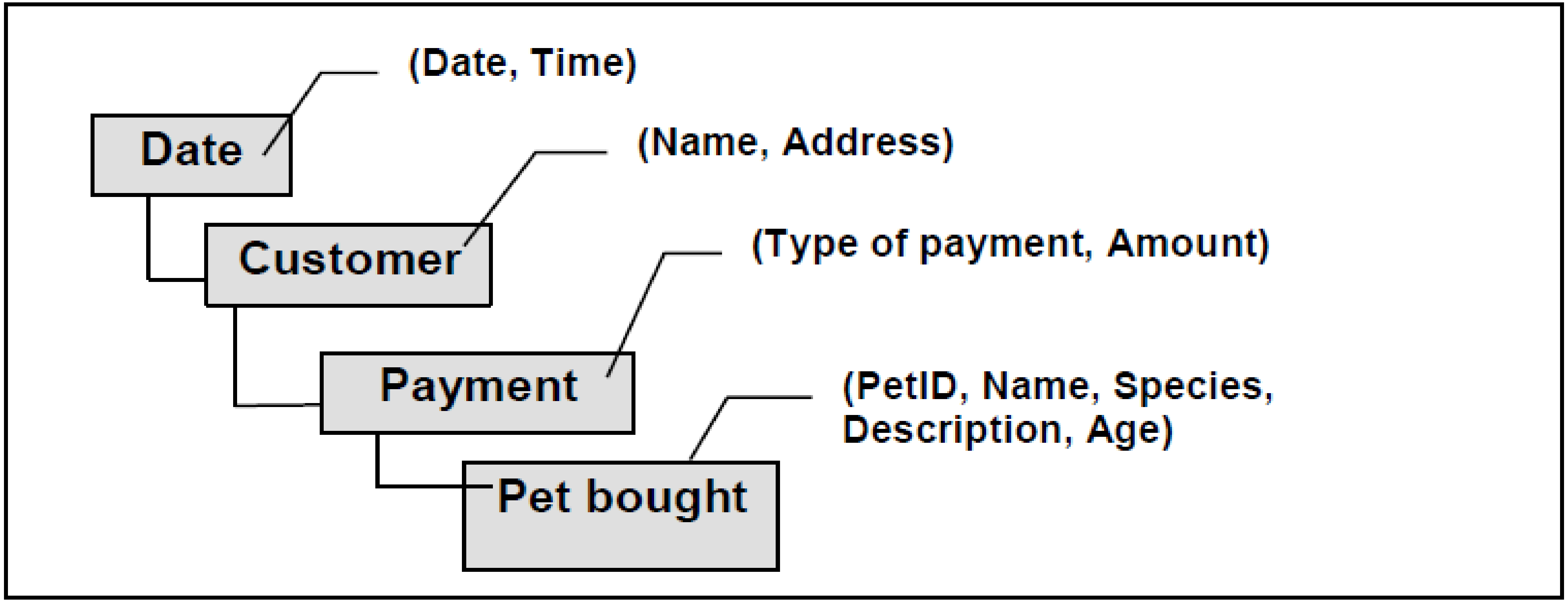
Data Model

- Depicts the logical organization of data in a database
- There are three types of data models
  - Hierarchical model
  - Network model
  - Relational model

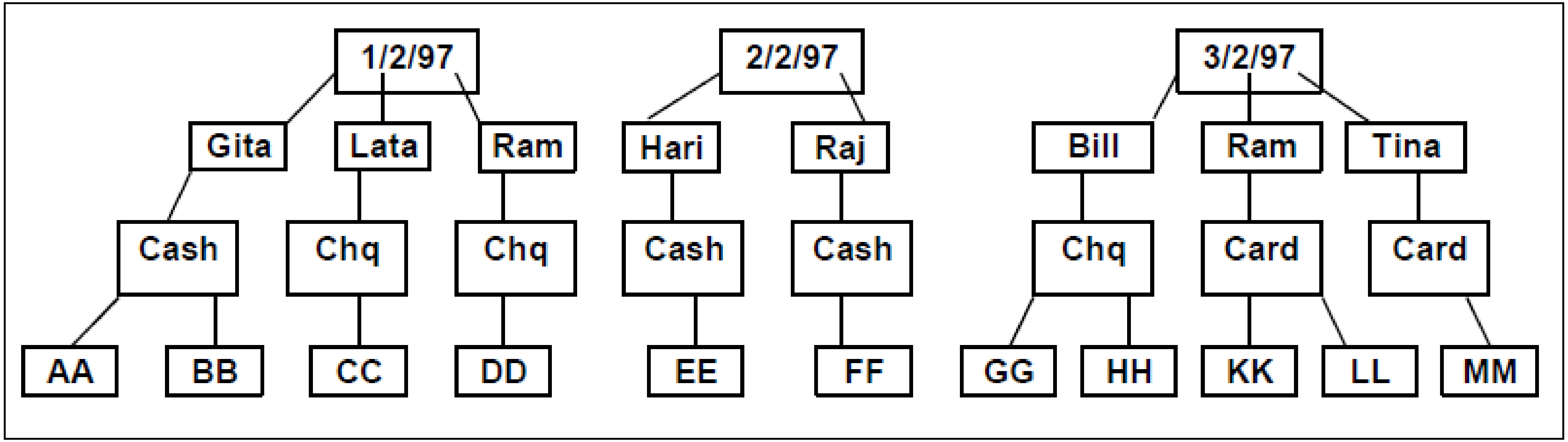
Hierarchical Model

- In the Hierarchical model data items are assigned to different levels of hierarchy
- Every data item (except the root node) acts a note with exactly one parent and zero of more children

Pet Store – Case Study



Hierarchical model for Pet Store



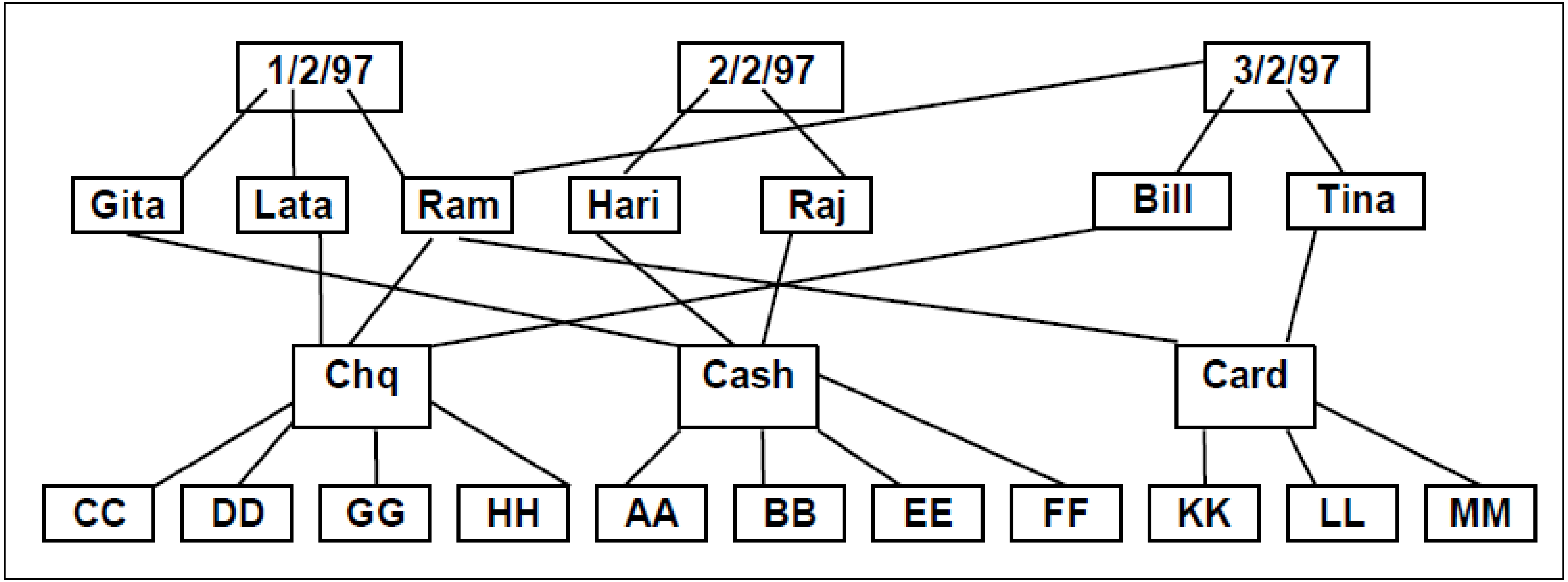
Issues with Hierarchical Model

- There is a lot of scope for duplication in this model. For example if the same customer buys a pet on four different days, then his address gets recorded four times
- Making modifications to a hierarchical database is extremely difficult. For example if the address of a customer has to be updated, we need to find out all possible occurrences of the address and update

Network Model

- Was proposed by CODASYL DBTG
- Eliminates the duplication of data items encountered in Hierarchical model
- Every data item appears only once as a node

Network Model for Pet Store



Issues with Network Model

- The degree of complexity is very high. Hence errors are very difficult to trace and fix
- Not flexible enough to change once data is entered

Relational Model

- Is based on the concepts of Relations in Set Theory
- Was triggered after the publication of a paper by E. F. Codd on the application of Relations to databases
- Represents the entire data in the form of tables and relations between those tables

Relational Model for Pet Store

CUSTOMER (CUST_NAME, ADDRESS)

PET (PET_ID, SPECIES, AGE, DATE, TIME)

PAYMENT (DATE, TIME, CUST_NAME, PAY_MODE, AMOUNT)

Relational Database Management System (RDBMS)

- Is a DBMS which manages Relational databases
- Advantages
  - Data Independence
  - Minimal duplication of data items
  - Promotes data sharing
  - Enforces data integrity
  - Provides data security
- Dis-advantages
  - One more layer between Application program and the data
  - Requires higher processing power from the system
  - Requires a large amount of memory

Data Modeling

- Is a technique used for Systems Analysis and Design
- Is a process that helps to reduce the gap between the customer knowledge and the analyst's interpretation
- Is the process of designing a data model from a set of customer requirements.
- Can be done in two ways
  - Bottom up approach (Normalization)
  - Top-down approach (ER-Modeling)

Entity Relationship (ER) Model & ER diagrams

- ER Model
  - Is a structured representation of entities, relationships, their properties and a detailed description of these
  - Consists of ER-diagrams & Supporting documentation
- ER Diagram
  - Represents entities, relationships between them and their significant properties
  - Is the central product of Entity analysis

Entity Type, Entity sub-type and Entity Instance

- An Entity type is a category of similar resources that are of interest in a given situation. Is represented by a rectangle in ERD
- Entity types can be divided into sub-types
- Example:
  - An entity type called Employee may be divided into sub-types such as Manager, Supervisor, Operator, etc.,
- An Entity instance is a thing that an enterprise recognizes as being capable of independent existence is uniquely identifiable
- Note: Unless otherwise mentioned Entity refers to Entity type.

Guidelines for naming Entity

- Use singular nouns
  - Example: EMPLOYEE and not EMPLOYEES
- Make it informative
  - Example: EMP-ADDR instead of ADDRESS
- Keep it concise
  - Example: EMP-ADDR instead of EMPLOYEE-ADDRESS

Tips of identifying Entities

- One should be able to instantiate it
- It should possess two or more attributes
- It should make business sense
- The thumb rule is identify the nouns in a given context and put them down as probable entities and eliminate the ones that obviously are not entities
- Note: Usually there exists more than one correct solution

Entity Examples

- PERSON: EMPLOYEE, STUDENT, TRAINER, CLIENT, ENGINEER, DEPENDANT, MANAGER, etc.,
- PLACE: CITY, OFFICE, REGION, STATION, SITE, BUILDING, SCHOOL, etc.,
- THING: PRODUCT, TOOL, PART, VEHICLE, PET, BOOK, etc.,
- CONCEPT: PROJECT, ORDER, ACCOUNT, COMPLAINT, BUSINESS-CYCLE, DEPARTMENT, LOAN, etc.,
- EVENT: PROJECT-PHASE, CHANGE-REQUEST, FUND-TRANSFER, PROMOTION, VACATION, etc.,

Attribute type and Attribute instance

- An attribute type is a single piece of information stored about an Entity type. Is represented by an ellipse in ERD
- An attribute instance is a specific data that can be stored about an Entity instance

EMPLOYEE

Name
Number
Office
Birth date
Eye colour

EMPLOYEE

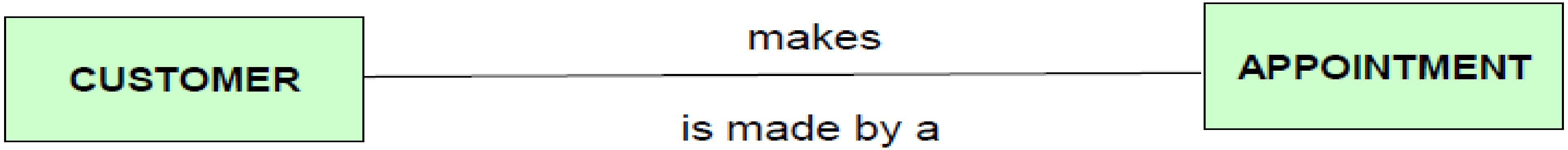
Rahul
3456
Infosys
12 June 1970
Brown

Exercise 1

- Categorize the following into Entity type, sub-type and instance
  - Furniture
  - Programmer
  - Sachin Tendulkar
  - Phone
  - Bangalore
  - Brigade Road
  - Mobile Phone
  - Home Loan

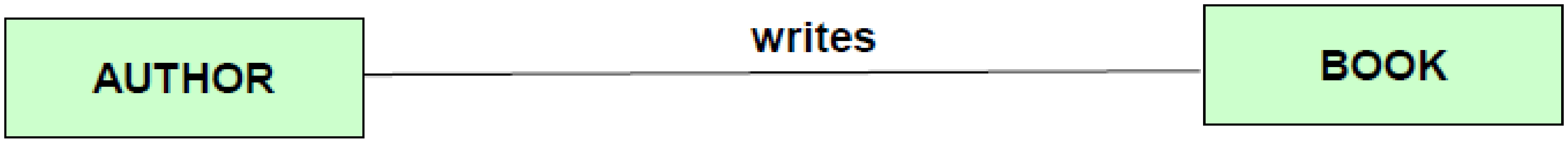
Relationship

- Is a named association between two or more entities. Is represented by a straight line in ERD
- Every relationship has a reciprocal relationship
- Example:
  - Customer makes an Appointment
  - An Appointment is made by a Customer



Naming Relationships

- Rule: Use a verb relating the two entities with a meaningful clause
- Guideline: Use an active verb. Example: writes, teaches, bills, etc.,

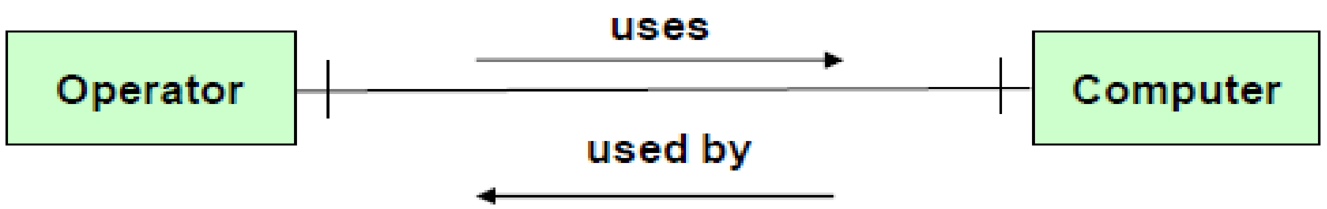


Degree and Cardinality

- Degree of a relationship: Specifies the number of entities participating in a relationship. Binary for 2, ternary for 3, quaternary for 4, etc.,
- Cardinality of relationship: Specifies the number of relationship instances that an entity can participate in.

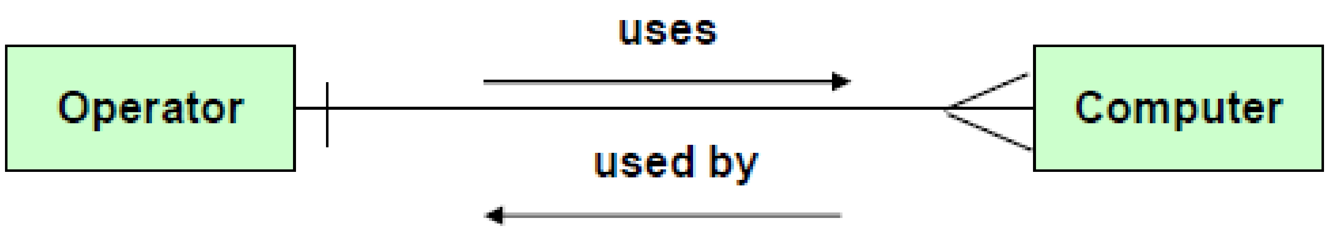
Types of Relationships (1 of 3)

- One to One
  - An Operator may use at most one Computer
  - A Computer may be used by at most one Operator



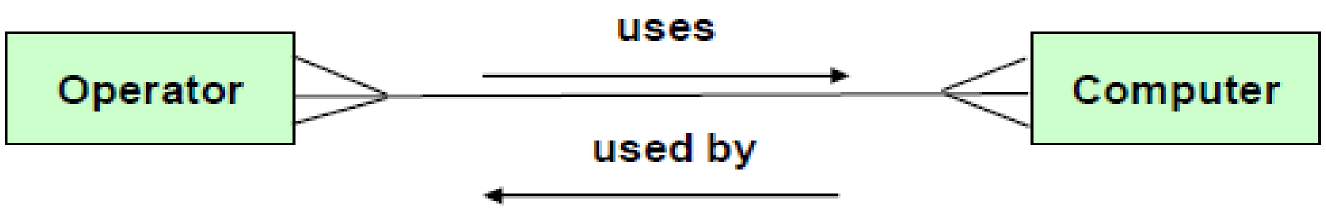
Types of Relationships (2 of 3)

- One to Many
  - An Operator may use many Computers
  - A Computer may be used by at most one Operator



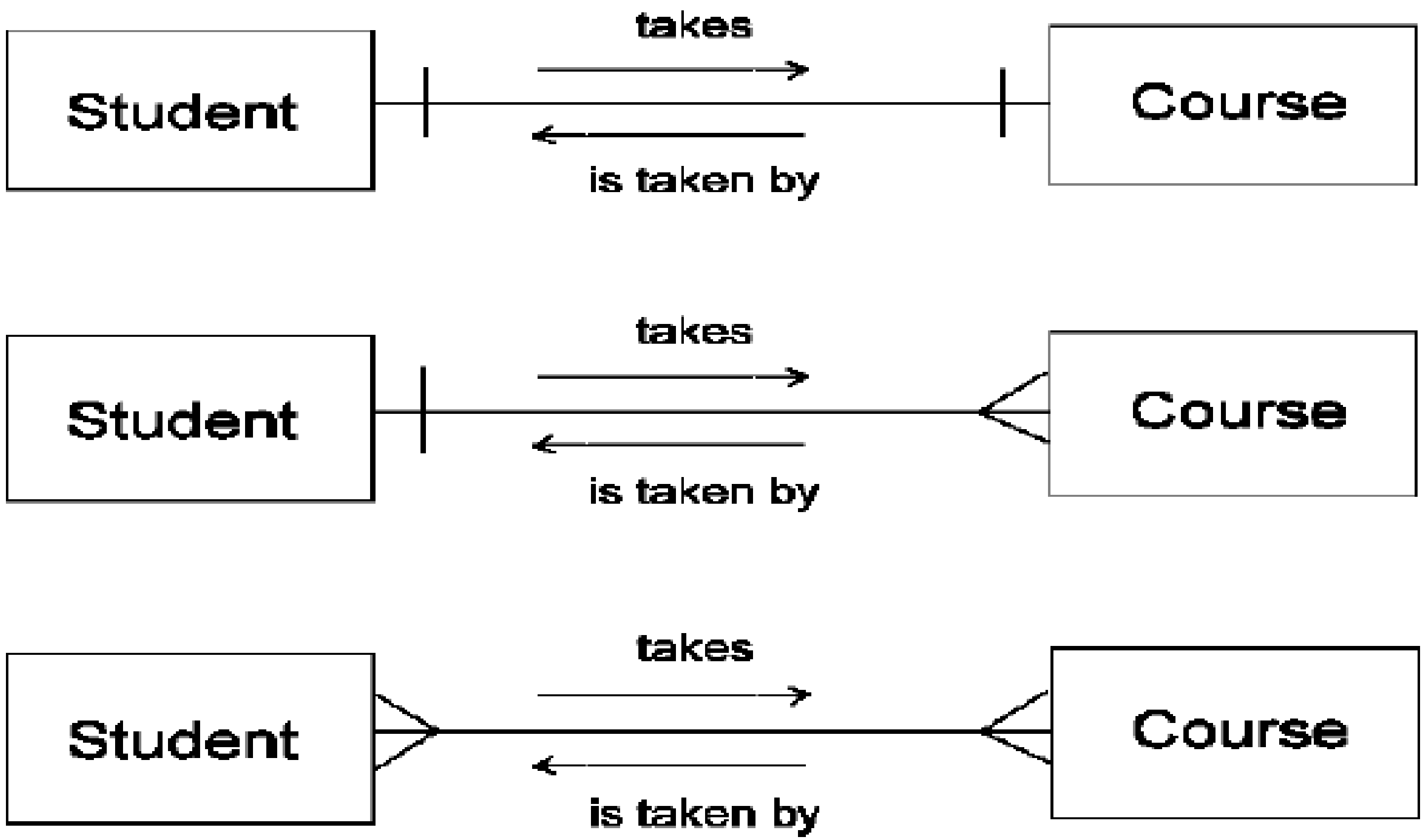
Types of Relationships (3 of 3)

- Many to Many
  - An Operator may use many Computers
  - A Computer may be used by many Operators

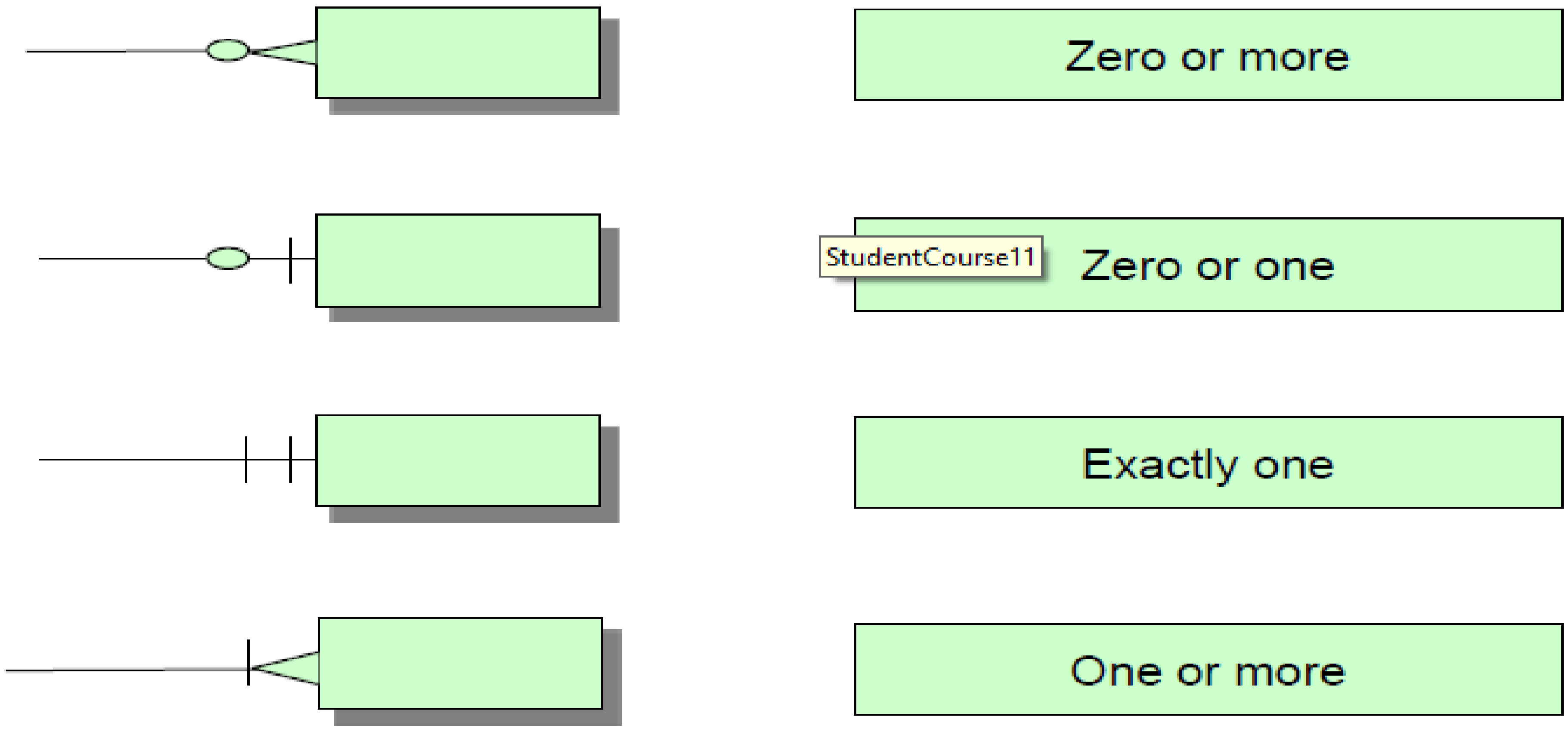


Exercise 2

- Write the relationship from the point of view of both entity types

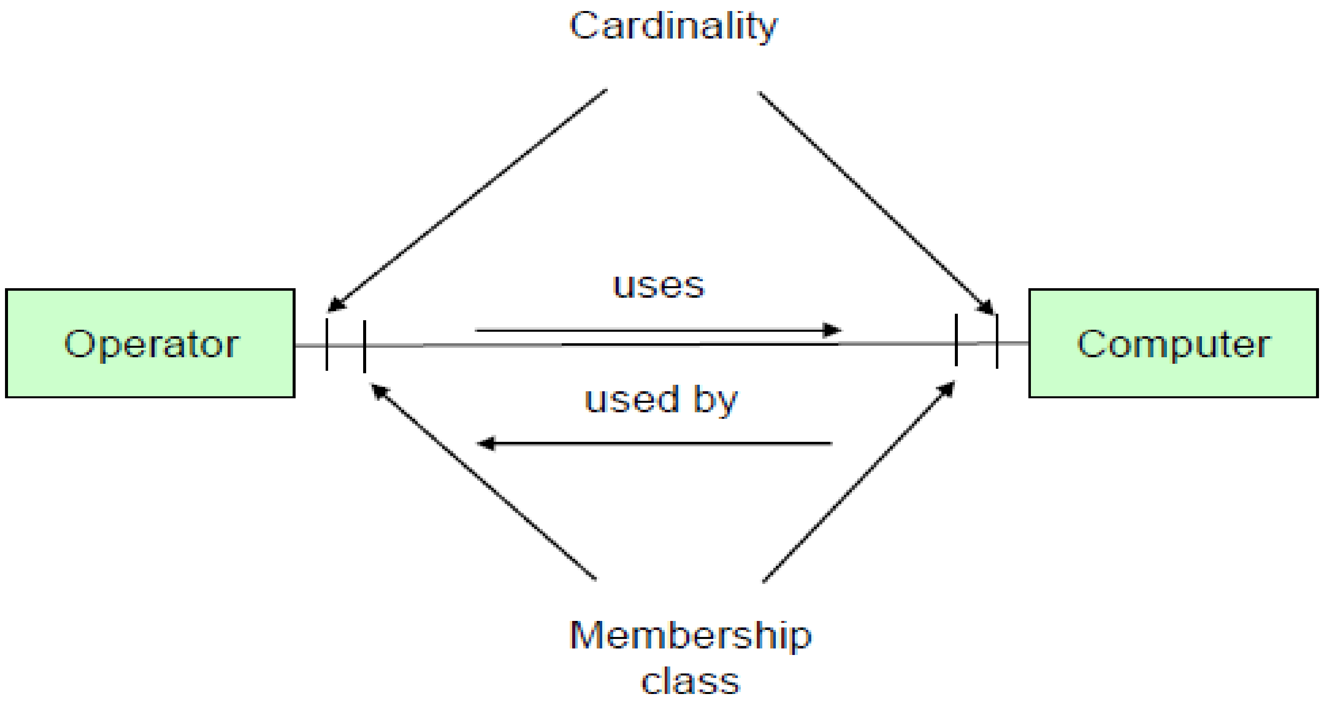


Membership class (1 of 4)



Membership class (2 of 4)

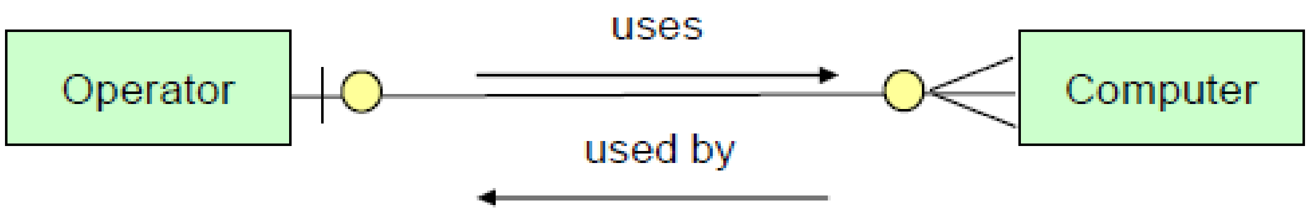
- Both sides obligatory



  - Every Operator uses exactly one Computer.
  - Every Computer is used by exactly one Operator.

Membership class (3 of 4)

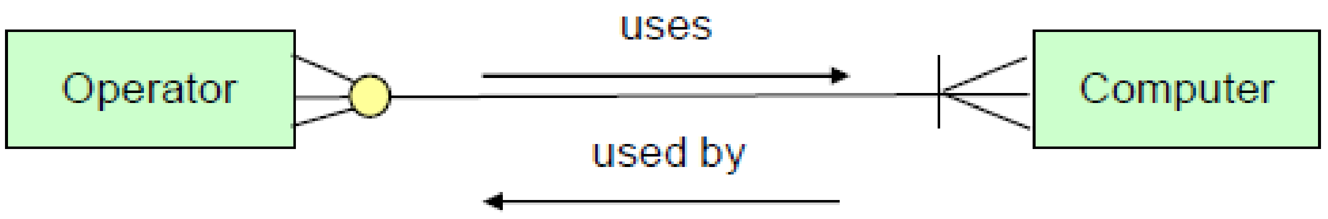
- Both sides non-obligatory



  - An Operator may use zero or more (any number of) Computers.
  - A Computer may be used by zero or one (at most one) Operator.

Membership class (4 of 4)

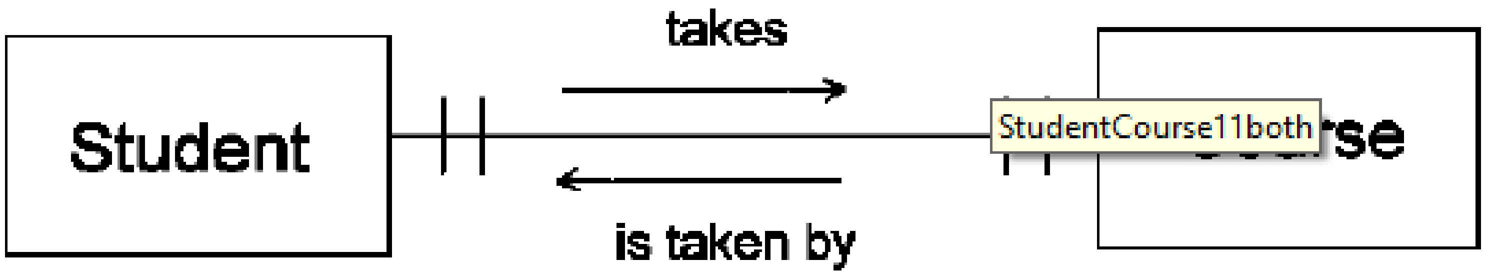
- One side obligatory



  - An Operator uses one or more Computers.
  - Every Computer may be used by zero or more Operators.

Exercise 3 (a)

- Write the membership of the following relationship from the point of view of both the entity types

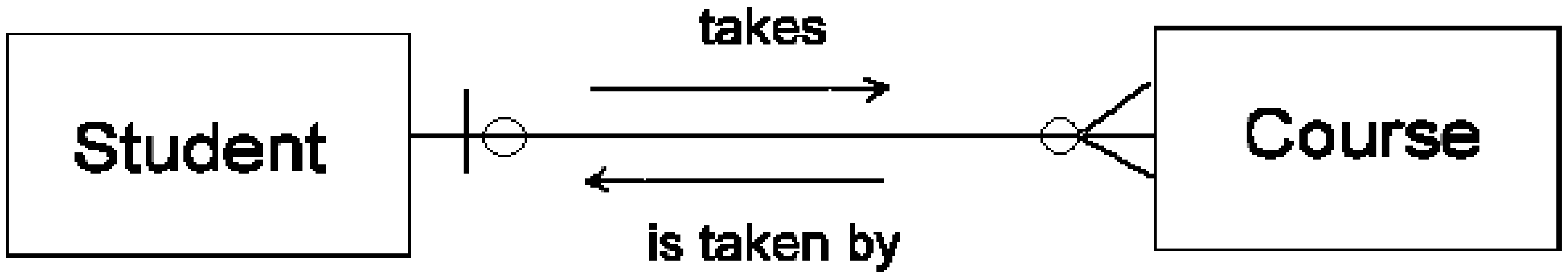


  – Ans (a) Every student takes exactly one course.

  – Ans (b) Every course is taken by exactly one student.

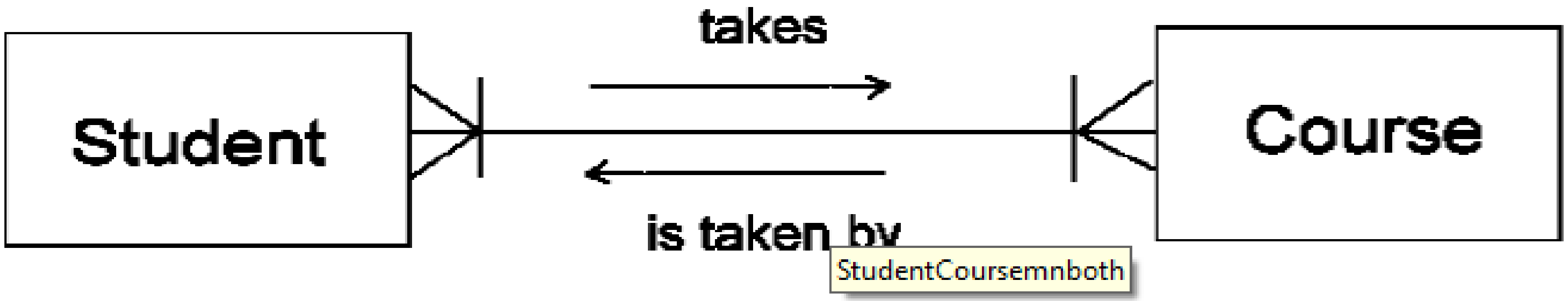
Exercise 3 (b)

- Write the membership of the following relationship from the point of view of both the entity types



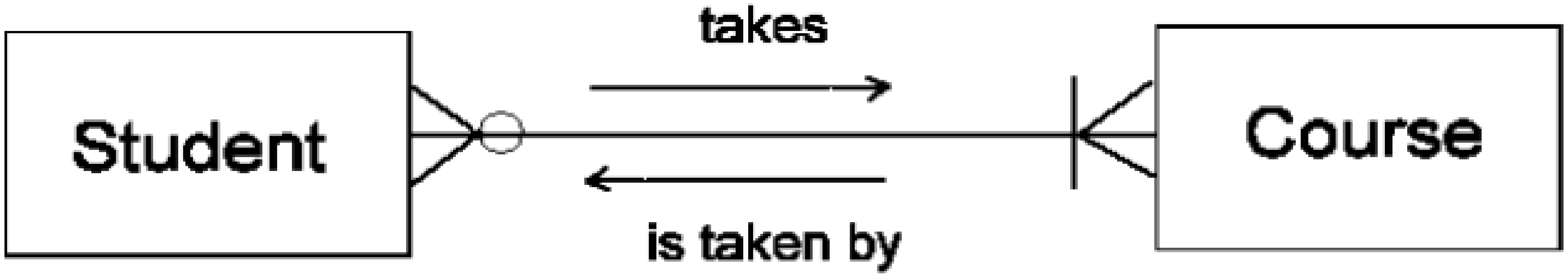
Exercise 3 (c)

- Write the membership of the following relationship from the point of view of both the entity types



Exercise 3 (d)

- Write the membership of the following relationship from the point of view of both the entity types



Case study 1

- An University contains many departments
- Each department offer several courses
- Each department employees several lecturers
- A lecturer can work only in one department
- For each department there is a Head
- A lecturer can be head of only one department
- Each lecturer teaches one or more of courses
- A course can be taught by only one lecturer
- A student can enroll for any number of courses
- Each course can have any number of students

Steps in ER Modeling

- Step 1: Identify the Entities
- Step 2: Identify the relationships and their cardinality
- Step 3: Identify the important attributes for every Entity
- Step 4: Draw the E-R diagram

Step 1

- DEPARTMENT
- STUDENT
- COURSE
- LECTURER

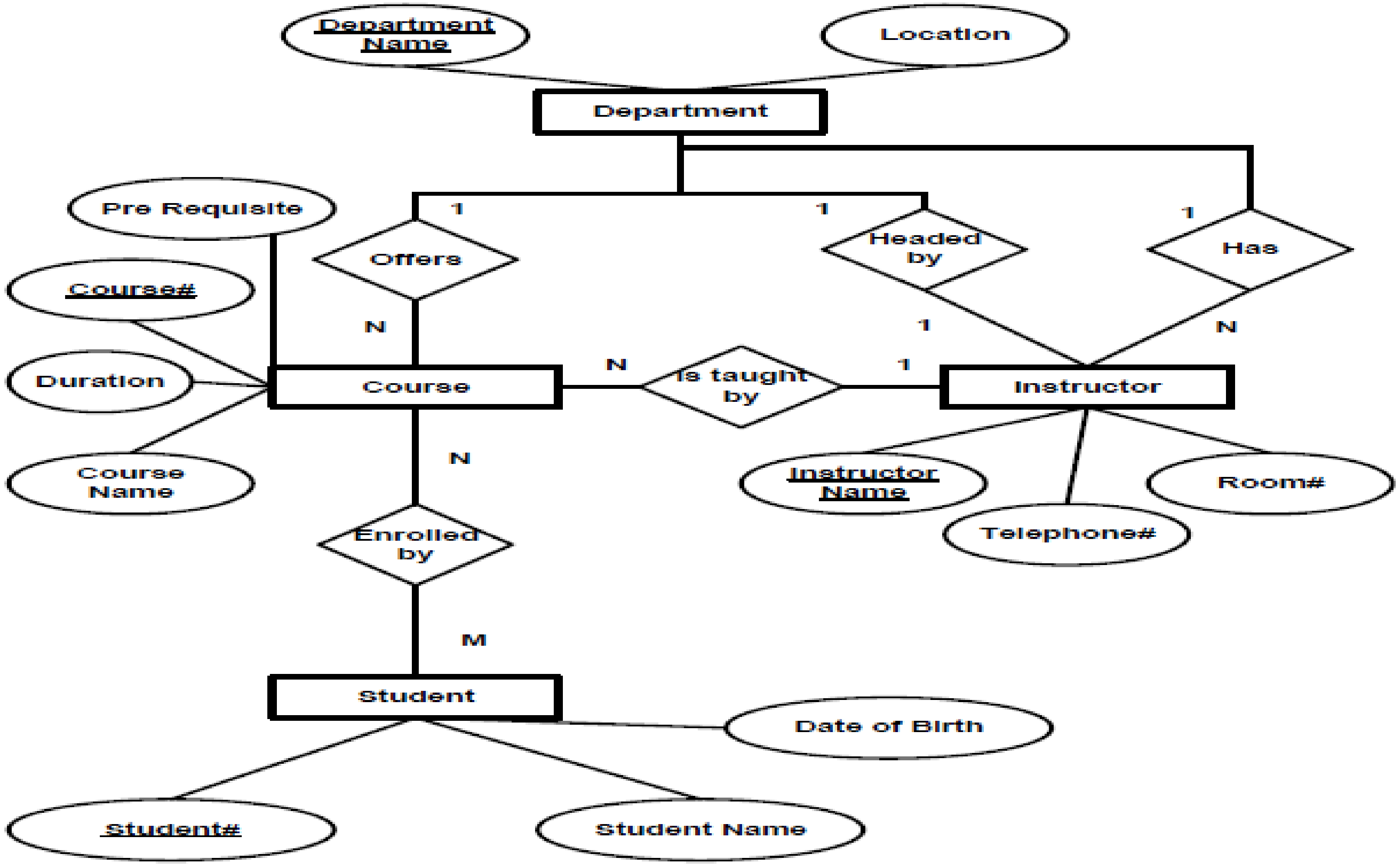
Step 2

- 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 lecturers and one lecturer belongs to one and only one department , hence the cardinality between department and lecturer is One to Many.
- Each department there is a "Head of department" and one lecturer is "Head of department ", hence the cardinality is One to One.
- One course is taught by only one lecturer, but the lecturer teaches many courses, hence the cardinality between course and lecturer is Many to One.

Step 3

- Department Name 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
- Lecturer Name is the key attribute for “lecturer” Entity

Step 4



Department Name
Location
Department
Pre Requisite
1
Offers
1
Headed by
1
Has
Course#
N
1
N
Duration
N
Is taught by
1
Course
Instructor
Course Name
N
Instructor Name
Room#
Telephone#
Enrolled by
M
Student
Date of Birth
Student#
Student Name

Case study 2

- There are multiple banks and each bank has many branches. Each branch has multiple customers
- Customers have various types of accounts
- Some Customers have also taken different types of loans from these bank branches
- One customer can have multiple accounts and Loans

Step 1

- BANK
- BRANCH
- LOAN
- ACCOUNT
- CUSTOMER

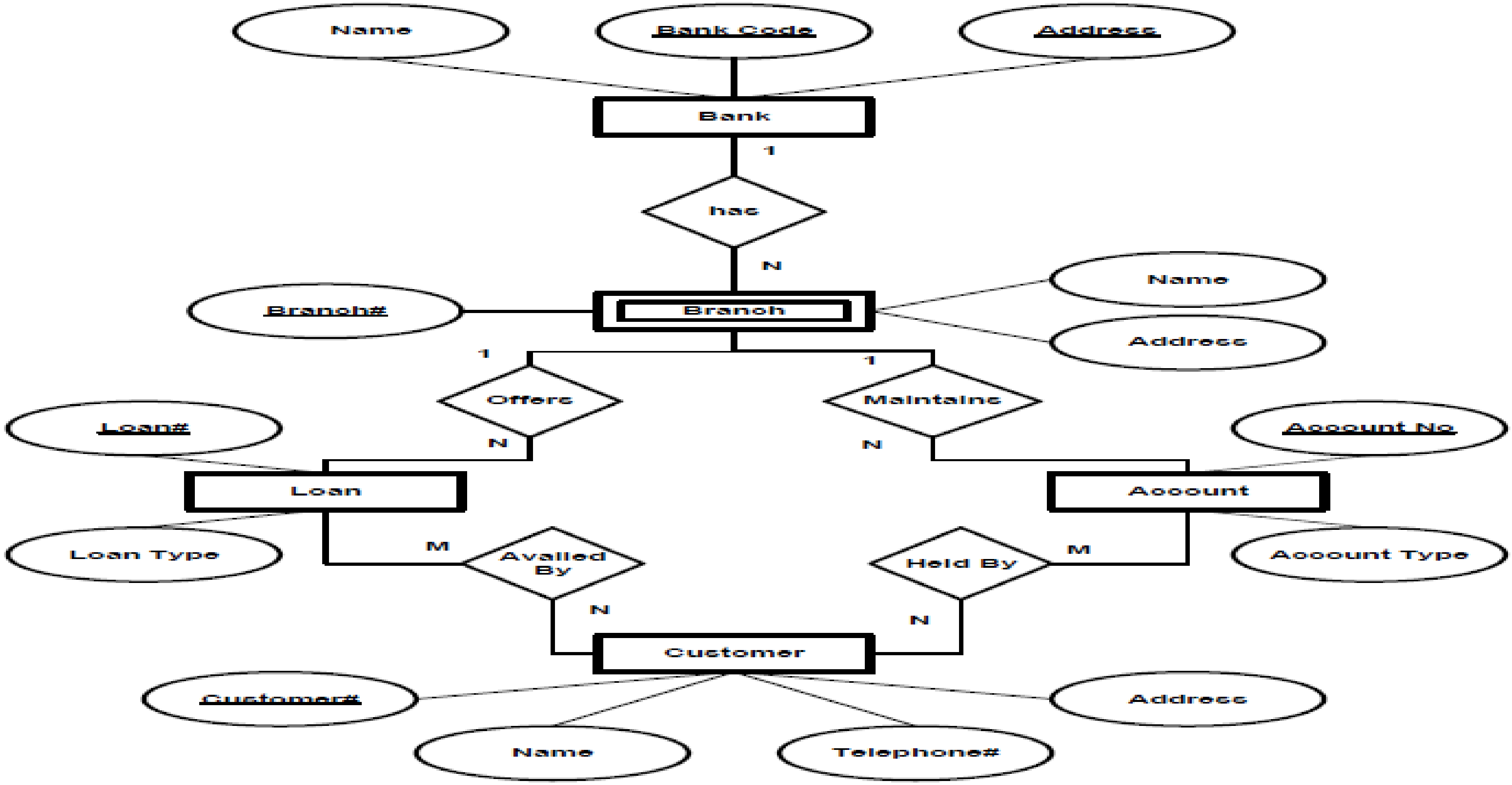
Step 2

- One Bank has many branches and each branch belongs to only one bank, hence the cardinality between Bank and Branch is One to Many
- One Branch offers many loans and each loan is associated with one branch, hence the cardinality between Branch and Loan is One to Many
- One Branch maintains multiple accounts and each account is associated to one and only one Branch, hence the cardinality between Branch and Account is One to Many
- One Loan can be availed by multiple customers, and each Customer can avail multiple loans, hence the cardinality between Loan and Customer is Many to Many
- One Customer can hold multiple accounts, and each Account can be held by multiple Customers, hence the cardinality between Customer and Account is Many to Many.

Step 3

- Bank Code (Bank Code) is the key attribute for the Entity “Bank”, as it identifies the bank uniquely
- Branch# (Branch Number) is the key attribute for “Branch” Entity
- Customer# (Customer Number) is the key attribute for “Customer” Entity
- Loan# (Loan Number) is the key attribute for “Loan” Entity
- Account No (Account Number) is the key attribute for “Account” Entity

Step 4



Name
Bank Code
Address
Bank
1
has
N
Branch#
Branch
Name
Address
1
Offers
1
Maintains
Loan#
N
N
Account No
Loan
Account
Loan Type
M
Availed By
Held By
M
Account Type
N
N
Customer
Customer#
Address
Name
Telephone#

Some Terminologies

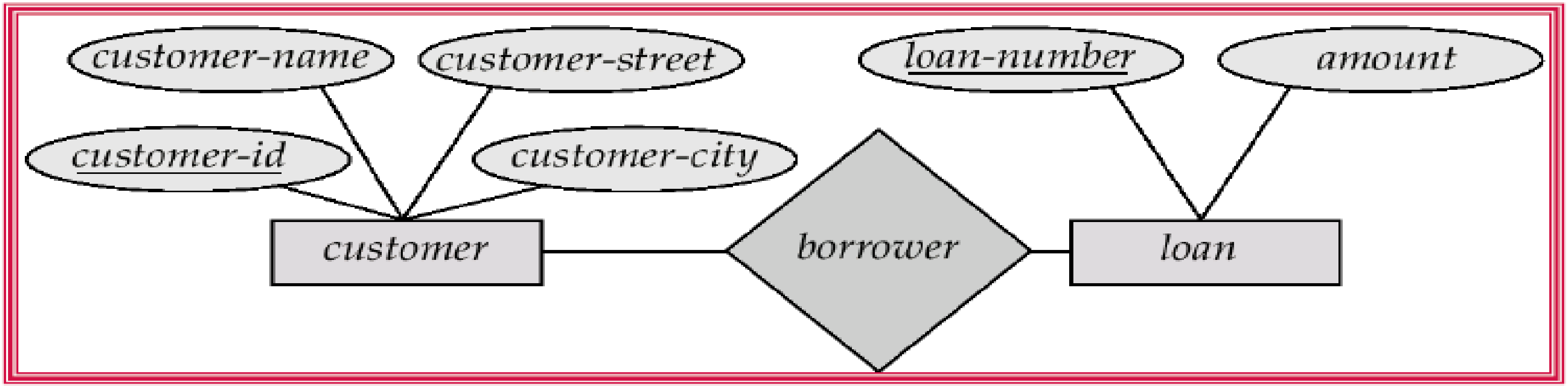
- Weak Entity
  - Is an entity which doesn't possess adequate number of attributes to form a candidate key
  - Example: Dependent for an Employee is a weak entity
- Multi-valued Attribute
  - Is an attribute that can have multiple values
  - Example: Telephone # is a multi-valued attribute
- Composite Attribute
  - Is a combination of two or more attributes
  - Examples: ADDRESS and DATE are composite attributes
- Derived Attribute
  - Is an attribute that can be derived some other attribute
  - Example: Grade is a derived attribute as it can be derived from the Marks attribute

N-ary Relationship

- Is a relationship between three or more entity sets
- Examples:
  - SUPPLIER supplies PART to PROJECT
  - DOCTOR prescribes MEDICINE to PATIENT

ERD - Notations

- Rectangle represent entity
- Rhombus represent relationship
- Ellipses represent attributes
  - Double ellipses represent multi-valued attributes
  - Dashed ellipses denote derived attributes
- Underline indicates primary key attributes (To be discussed)



Relational Model

- Is a data model in which the entire data is represented using only tables and relation between tables
- The rows and columns of a table are also referred to as tuples and attributes respectively
- The number of rows and columns in a table are referred to as its cardinality and degree respectively

Candidate key

- Is a set of one or more columns of the table whose combined value is unique in the table
- Must be irreducible. i.e., no proper subset of a candidate key should possess the uniqueness property
- Example: Consider the following table. Here, both EMPNO and MAILID are candidate keys

EMPNO	EMP_NAME	DEPTNO	MAILID

- Note: Any superset of a candidate key is called a Super key.

Primary key

- Is one of the candidate keys
- Criteria for choosing primary key
  - Minimal of the candidate keys
  - Business rules
- In the Unisys context, EMPNO is better choice for primary key than MAILID. Why?

EMPNO	EMP_NAME	DEPTNO	MAILID

Foreign key

- Is a column in a table which is the primary key column in some other table
- Example: In the following example DEPTNO is the primary key in the DEPARTMENT table and hence is the foreign key in the EMPLOYEE table

EMPNO	EMP_NAME	DEPTNO	MAILID

DEPTNO	DEPT_NAME

Foreign key – Rules (Referential Integrity)

- ON DELETE CASCADE
- ON DELETE RESTRICT
- ON DELETE SET NULL
- ON DELETE SET DEFAULT

Composite key and Overlapping key

- A candidate key comprising of two or more columns is called a composite key
- Two or more candidate keys having one or more common columns are called overlapping keys
- Note: The columns that are not a part of any of the candidate key are called non-key columns

Converting ERD to Relational model (1 of 2)

- Each entity type is converted into a table
- Each single-valued attribute is made a column in the corresponding table
- Derived attributes are ignored
- Multi-valued attributes are represented by a separate table
- Weak entities are converted into a table of their own, with the primary key of the strong entity acting as a foreign key in the table. This foreign key along with the key of the weak entity form the composite primary key of this table

Converting ERD to Relational model (1 of 2)

- Each entity type is converted into a table
- Each single-valued attribute is made a column in the corresponding table
- Derived attributes are ignored
- Multi-valued attributes are represented by a separate table
- Weak entities are converted into a table of their own, with the primary key of the strong entity acting as a foreign key in the table. This foreign key along with the key of the weak entity form the composite primary key of this table

Functional dependence

- Given a table T, a column Y of T is said to be functionally dependent on a column X of T (denoted by $X \rightarrow Y$), provided for every value in column X there corresponds a unique value in column Y
- Example: In the following table EMPNAME, DEPTNO and MAILID are all functionally dependent on EMPNO, since for every value of EMPNO there exists one value of EMPNAME, DEPTNO and MAILID

EMPNO	EMP_NAME	DEPTNO	MAILID

- There are 3 types of functional dependence viz.,
  - Full dependence
  - Partial dependence
  - Transitive dependence

Full dependence

- A column Y of a table T is said to be fully dependent on a combination of columns X of T, provided it is functionally dependent on X & not functionally dependent on any proper subset of A
- Example: In the following example the marks is fully dependent on the combination of columns STUD# and COURSE #

STUD #	COURSE #	STUD_NAME	COURSE_TITLE	MARKS	GRADE

Partial dependence

- A column Y of a table T is said to be partially dependent on a combination of columns X of T, provided it is functionally dependent on X and at least one proper subset of A
- Example: In the following example COURSE_TITLE is partially dependent on the composite key (Stud # , Course #)

STUD #	COURSE #	STUD_NAME	COURSE_TITLE	MARKS	GRADE

Transitive dependence

- A column Y of a table T is said to be transitively dependent on a column X of T, provided it is functionally dependent on a column Z of T which in turn is functionally dependent on X

- Example: In the following example GRADE is transitively dependent on the composite key (STUD # , COURSE #), since GRADE is dependent on MARKS which in turn is functionally dependent on the key (STUD # , COURSE #)

STUD #	COURSE #	STUD NAME	COURSE TITLE	MARKS	GRADE

First Level design of Database (Normalization)

- Is a process used to remove the ambiguities and inconsistencies that may exist in the schema derived from ERDs
- The following are the characteristic features of Normalized tables
  - No data should be duplicated in different rows unnecessarily
  - The intersection of every row and every column should contain some entry
  - If a row is added to a table, then the existing rows in the table as well as other tables in the database must be unaffected
  - If a row is deleted from a table, then important information shouldn't be lost
  - Any row of a table can be updated independent of other rows in the table

Un-normalized Table

S#	Name	Status	City	P#	Color	Weight	Quantity
S1	Smith	20	London	P1 P2 P3	Red Blue Orange	12 17 17	300 200 400
S2	Jones	10	Paris	P1 P2	Red Orange	12 17	100 400
S3	Clark	30	Rome	P2	Blue	17	500
S4	Adam	20	London	P4	Grey	15	300

1st Normal Form

- A table is said to be in the 1st Normal Form provided the intersection of every row and column contain only atomic values

S#	Name	Status	City	P#	Color	Weight	Quantity
S1	Smith	20	London	P1	Red	12	300
S1	Smith	20	London	P2	Blue	17	200
S1	Smith	20	London	P3	Orange	17	400
S2	Jones	10	Paris	P1	Red	12	100
S2	Jones	10	Paris	P2	Orange	17	400
S3	Clark	30	Rome	P2	Blue	17	500
S4	Adam	20	London	P4	Grey	15	300

- Note: The primary key for this table can be SNO and PNO or SNAME and PNO

Issues with 1st Normal Form

- Insertion
  - A subset of a row cannot be inserted.
- Deletion
  - Valuable information may be lost in the process.
- Updation
  - May have to be repeated for the same piece of information.

2nd Normal Form (1 of 2)

- A Table is said to be in the 2nd Normal Form provided it is in first normal form and every non-key column is fully functionally dependent on candidate key column
- In order to reduce a table to 2nd NF, create a separate table containing the column that is partially dependent on the candidate key and that subset of candidate key on which it depends
- Note: The 2 NF may contain transitive dependencies.

2^{nd} Normal Form (2 of 2)

S#	Name	Status	City	P#	Color	Weight	Quantity
S1	Smith	20	London	P1	Red	12	300
S1	Smith	20	London	P2	Blue	17	200
S1	Smith	20	London	P3	Orange	17	400
S2	Jones	10	Paris	P1	Red	12	100
S2	Jones	10	Paris	P2	Orange	17	400
S3	Clark	30	Rome	P2	Blue	17	500
S4	Adam	20	London	P4	Grey	15	300

S#	Name	Status	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Clark	30	Rome
S4	Adam	20	London

S#	P#	Color	Weight	Quantity
S1	P1	Red	12	300
S1	P2	Blue	17	200
S1	P3	Orange	17	400
S2	P1	Red	12	100
S2	P2	Orange	17	400
S3	P2	Blue	17	500
S4	P4	Grey	15	300

Issues with 2nd Normal Form

- Since a table in 2nd NF may have transitive dependencies, it may contain two or more non-key columns that are functionally dependent. This will result in unnecessary duplication of data

- Example: In the SP table the STATUS column is dependent on CITY.

3rd Normal Form (1 of 2)

- A Table is said to be in the 3rd Normal Form provided it is in second normal form and every non-key column is non-transitively dependent on candidate key
- In order to reduce a table to 3 NF create a separate table containing the columns that are functionally dependent

3rd Normal Form (2 of 2)

S#	Name	Status	City	P#	Color	Weight	Quantity
S1	Smith	20	London	P1	Red	12	300
S1	Smith	20	London	P2	Blue	17	200
S1	Smith	20	London	P3	Orange	17	400
S2	Jones	10	Paris	P1	Red	12	100
S2	Jones	10	Paris	P2	Orange	17	400
S3	Clark	30	Rome	P2	Blue	17	500
S4	Adam	20	London	P4	Grey	15	300

S#	Name	Status
S1	Smith	20
S2	Jones	10
S3	Clark	30
S4	Adam	20

Status	City
20	London
10	Paris
30	Rome

S#	P#	Color	Weight	Quantity
S1	P1	Red	12	300
S1	P2	Blue	17	200
S1	P3	Orange	17	400
S2	P1	Red	12	100
S2	P2	Orange	17	400
S3	P2	Blue	17	500
S4	P4	Grey	15	300

Boyce-Codd Normal Form (BCNF)

- A Table is said to be in the Boyce-Codd Normal Form provided it is in third normal form and there are no-overlapping candidate keys
- Note: The BCNF is also referred as a Strong 3 NF.

Summary of Normal Forms

Input	Operations	Output
Un-normalized Table	Create separate rows for every combination of multi-valued columns	Table in 1 NF
Table in 1 NF	Eliminate Partial dependencies on the candidate key	Tables in 2NF
Tables in 2 NF	Eliminate Transitive dependencies	Tables in 3 NF
Tables in 3 NF	Eliminate Overlapping candidate key columns	Tables in BCNF

Note

- There is nothing like an ideal normal form
- Full normalization will have adverse effect in retrieval and hence is not always desirable
- Use your discretion based on the situation

Second level design of Database

- Is an essential part of database design
- Involves selective introduction of duplication to meet specific performance requirements
- There are 4 techniques used in the second level design
  - Elimination of tables
  - Merging of tables
  - Store derived attributes
  - Horizontal partitioning of tables (creation of Entity sub-types)