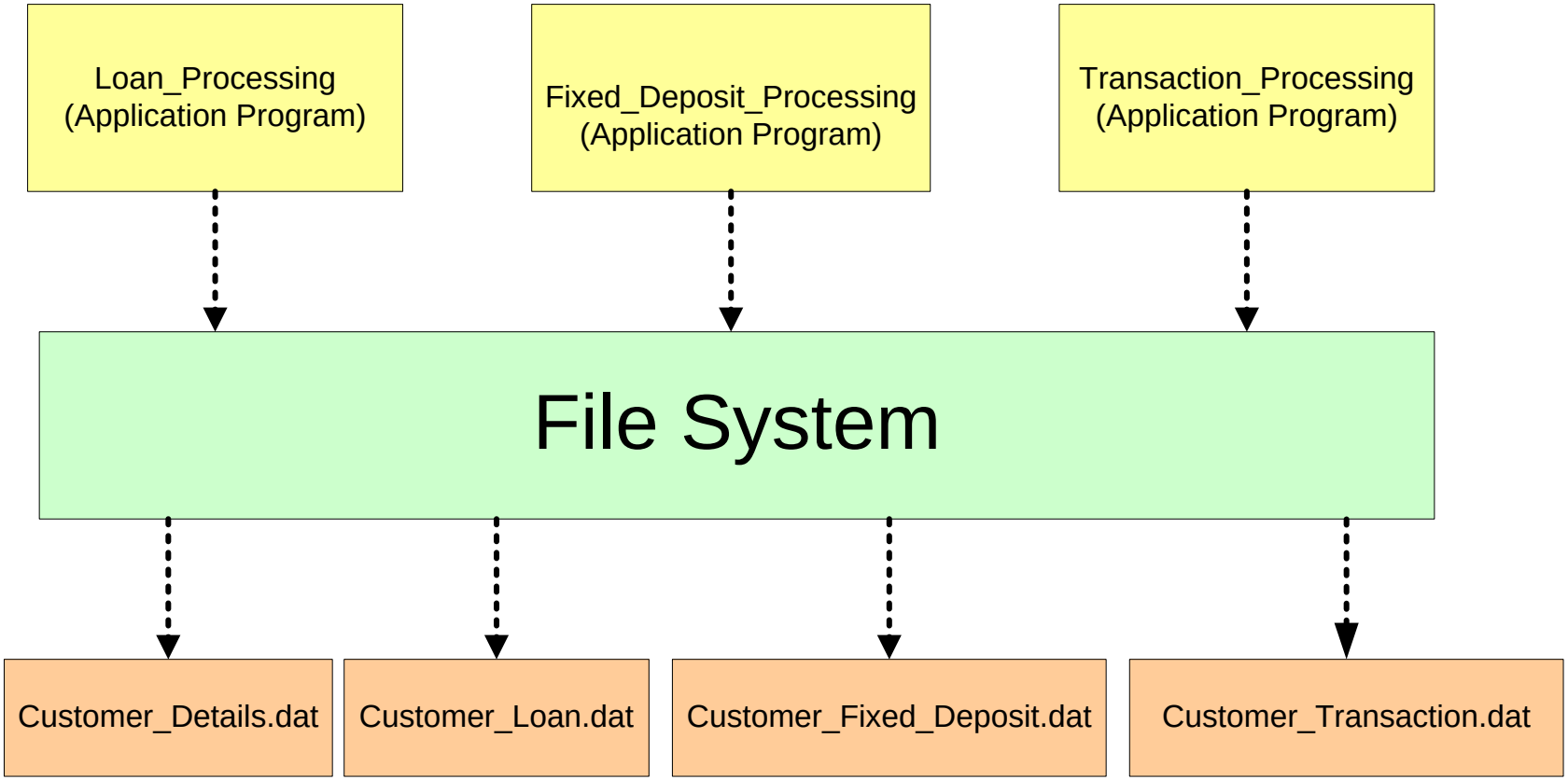
RDBMS

Traditional Method of Data Storage



Ways of storing data in files – Customer Data

Data used to be stored in the form of records in the files.

Records consist of various fields which are delimited by a space , comma , tab etc.

There used to be special characters to mark end of records and end of files.

```
1011  Ravi Kumar       10
1012  Laxmi            20
1013  Krishna Prasad   30
1014  Srikanth Reddy    10
1015  Vivek            20
```

Problems:Traditional approach

- Data Security
- Data Redundancy
- Data Isolation
- Program/Data Dependence
- Lack of Flexibility
- Concurrent Access Anomalies

The Database Technology

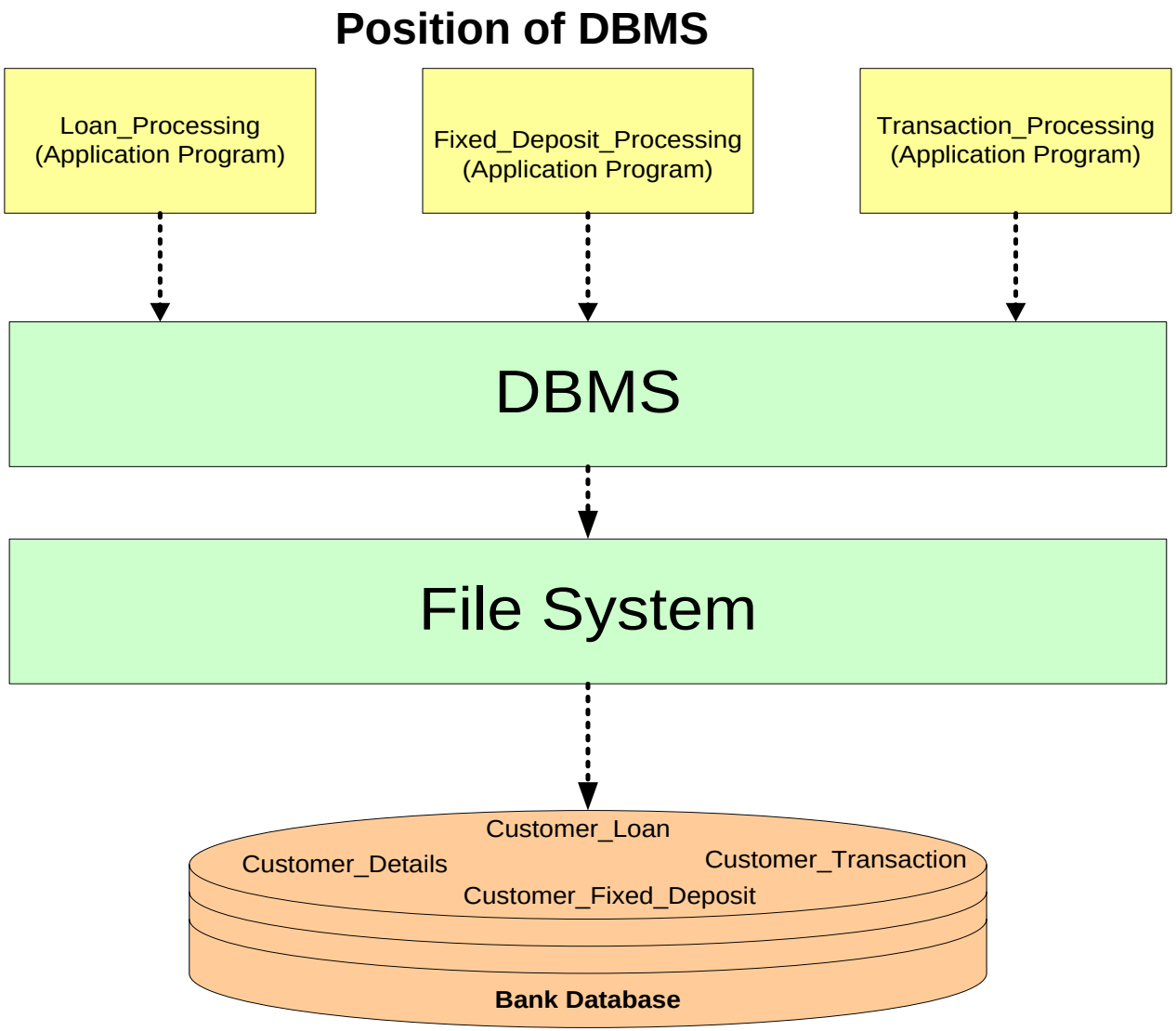
Database

- Computer based record-keeping system
- Organized collection of interrelated (persistent) data
- Records & maintains data

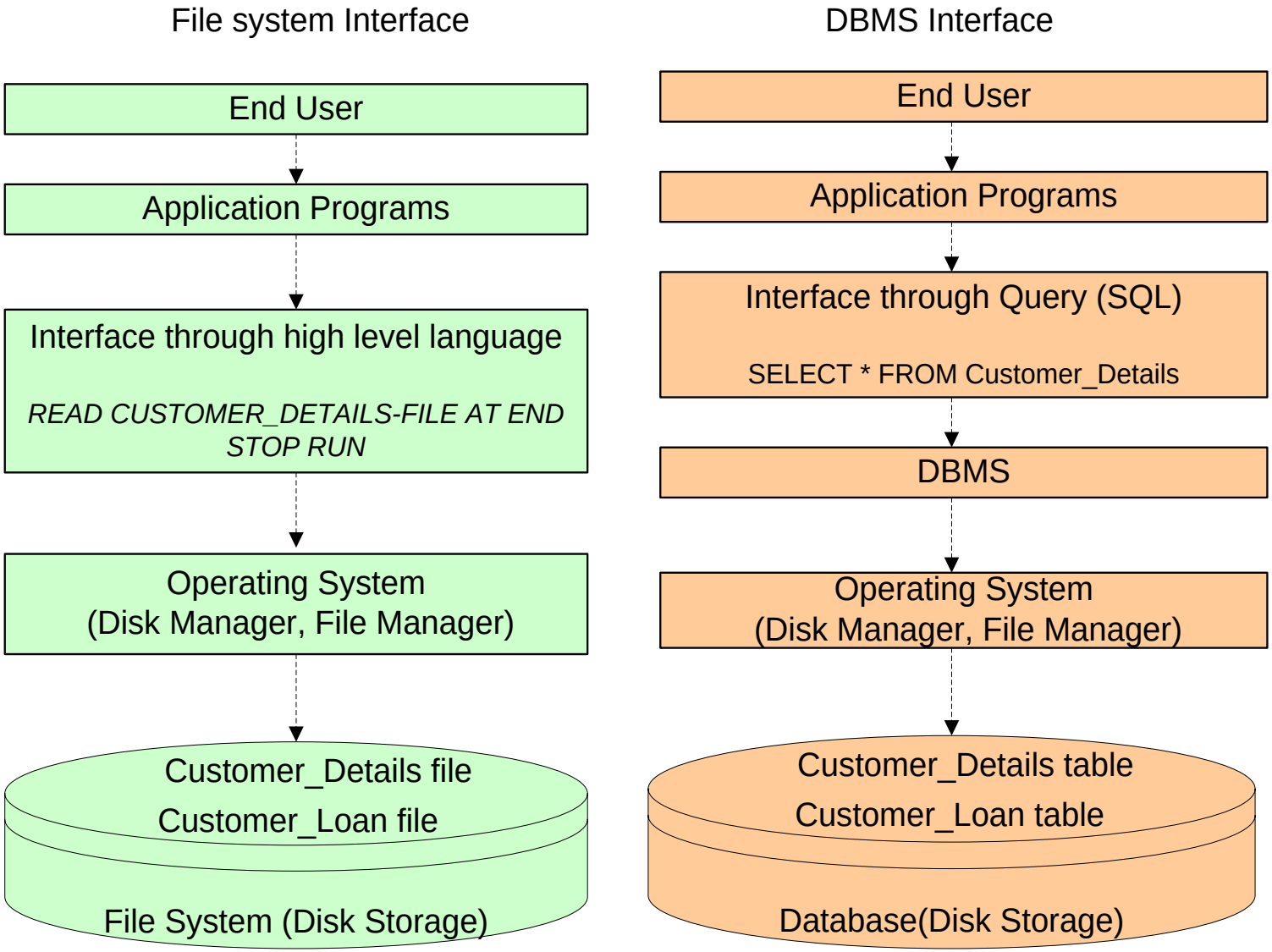
Database Management System

- 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

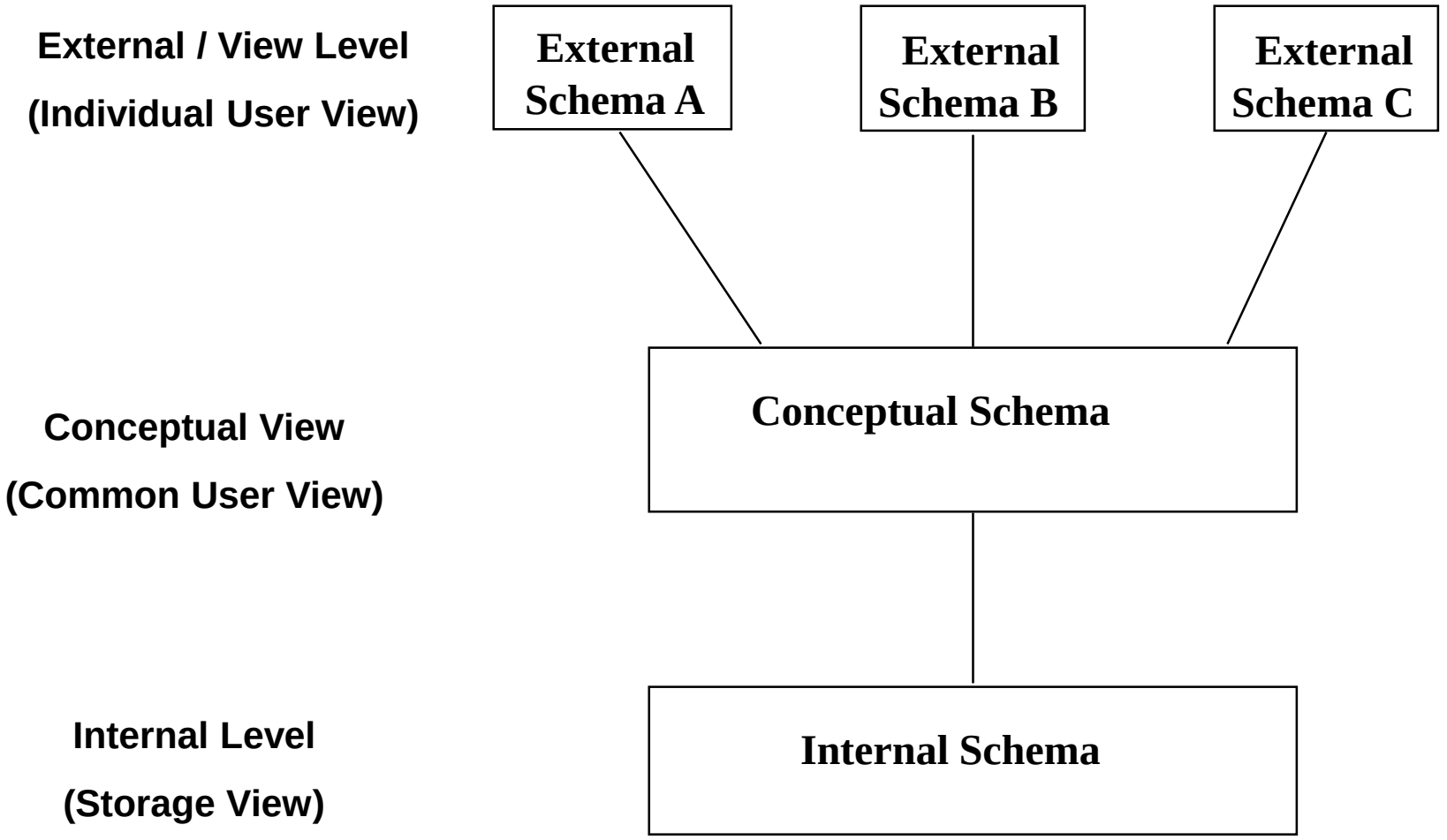
Where does the DBMS fit in?



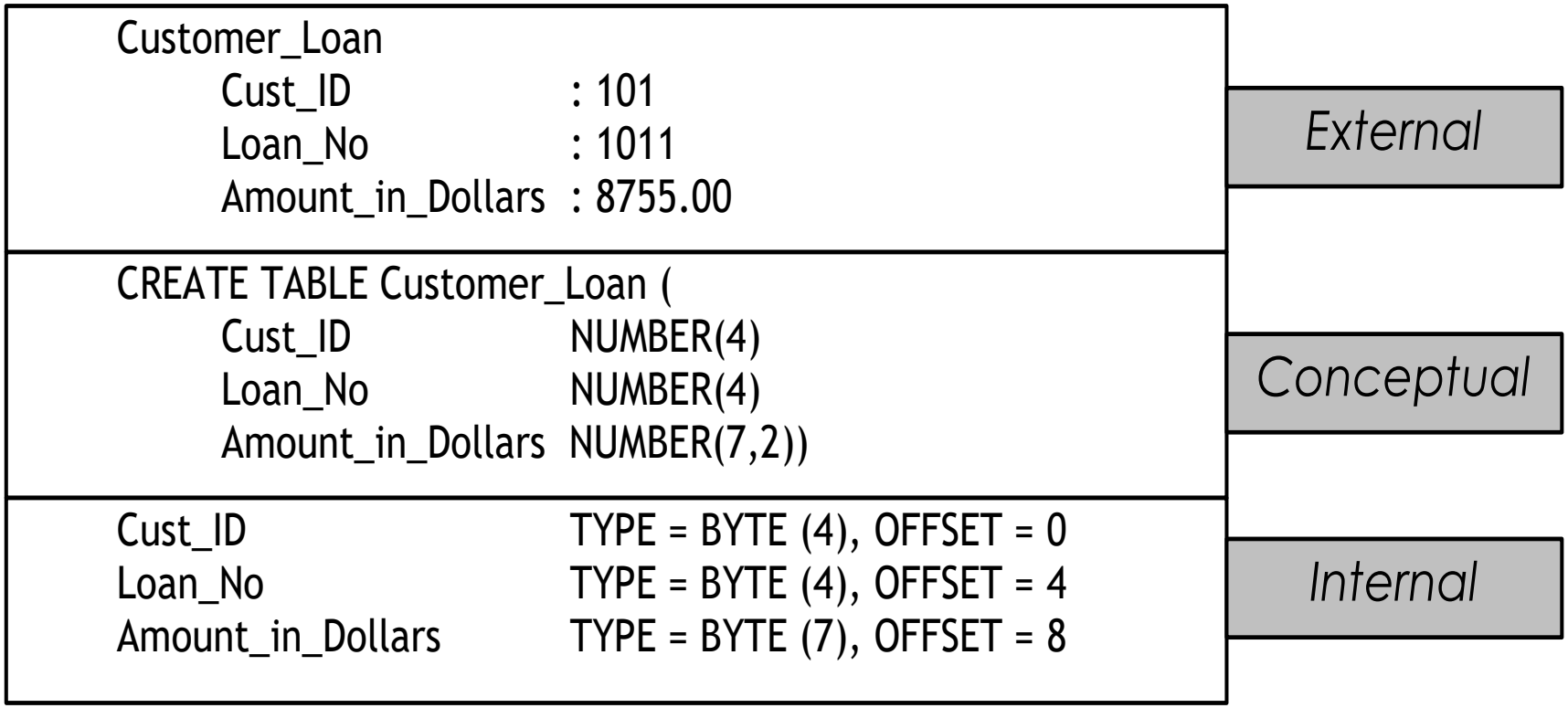
Difference Between File and DBMS Operations



Three-layer Architecture



An example of the three levels



Users of a DBMS

- Database Administrator (DBA)
  - Managing information contents
  - Liaison with users
  - Enforcing security and integrity rules
  - Strategizing backup & recovery
  - Monitoring performance

- Database designers

- Application Programmers

- End users

Advantages of a DBMS

- Data independence
- Reduction in data redundancy
- Better security
- Better flexibility
- Effective data sharing
- Enforces Integrity Constraints
- Enables Backup and recovery

Data Models

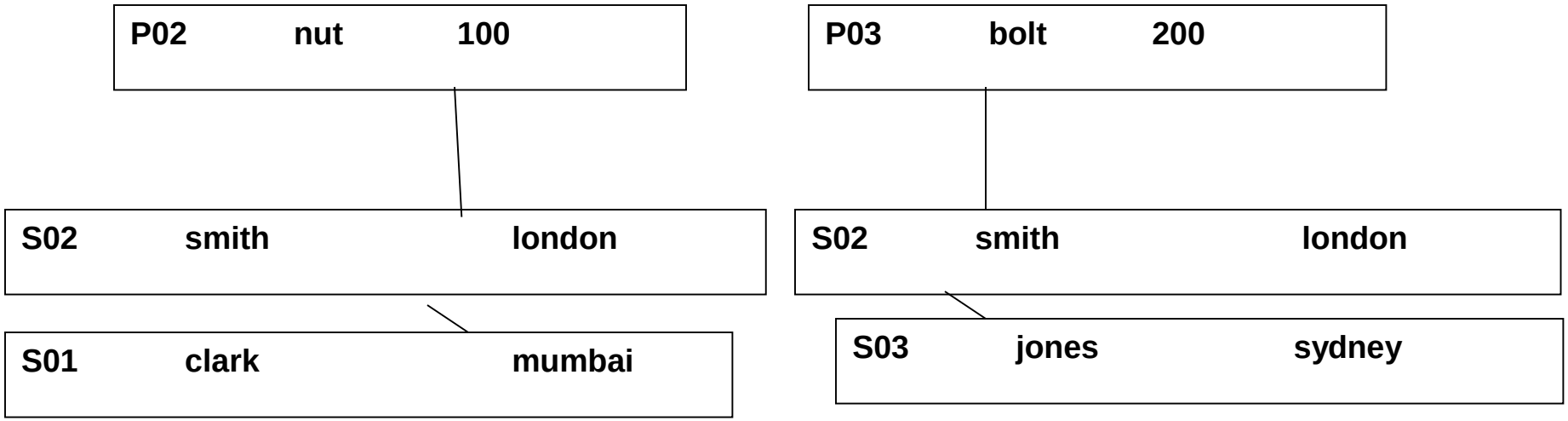
Data Model :

- A Conceptual tool used to describe
  - Data
  - Data Relationships
  - Data Semantics
  - Consistency Constraints

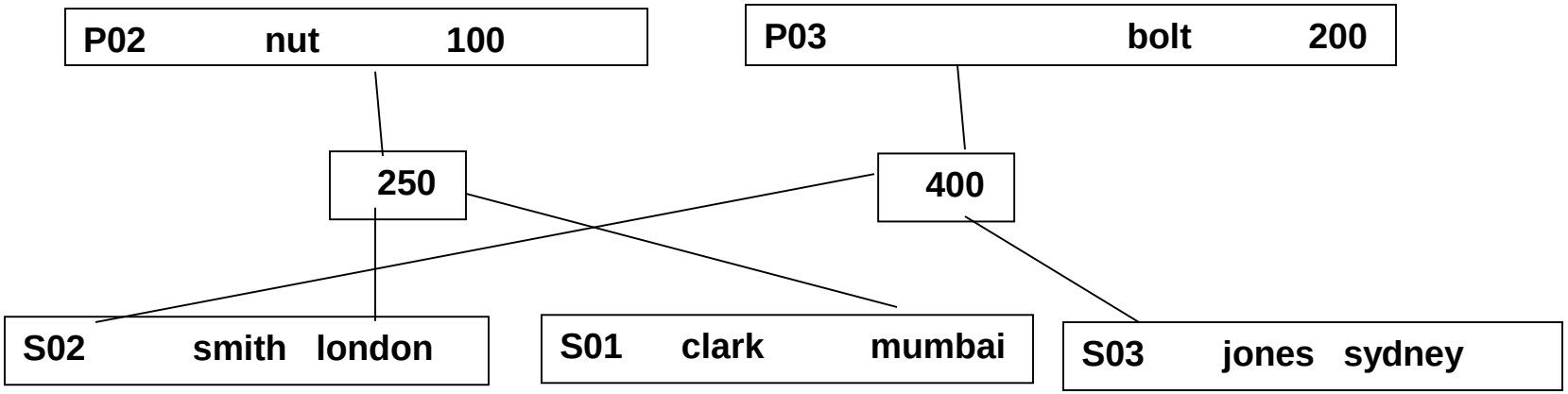
Types of Data Models

- Object Based Logical Model
  - Entity Relationship Model
- Record Based Logical Model
  - Hierarchical Data Model
  - Network Data Model
  - Relational Data Model

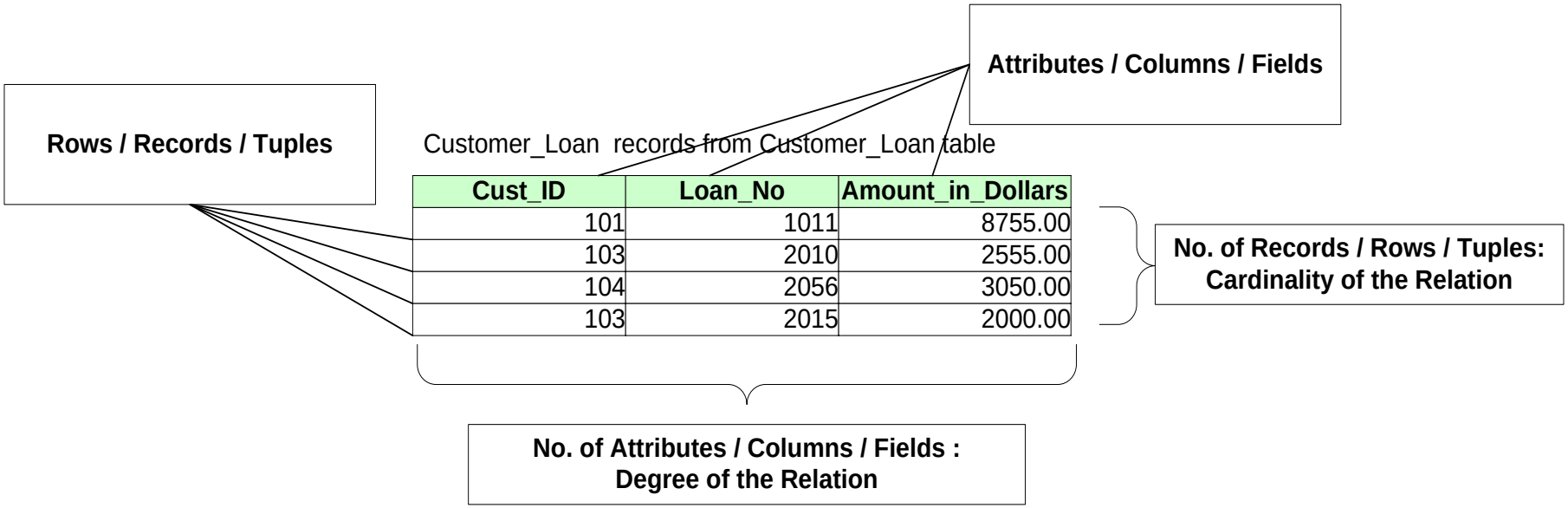
Record Based Data Model – Hierarchical Data Model



Record Based Data Model – Network Data Model



Record Based Data Model – Relational Data Model



Cust_ID	Loan_No	Amount_in_Dollars
101	1011	8755.00
103	2010	2555.00
104	2056	3050.00
103	2015	2000.00

Cust_ID	Cust_Last_Name	Cust_Mid_Name	Cust_First_Name	Account_No	Account_Type	Bank_Branch	Cust_Email
101	Smith	A.	Mike	1020	Savings	Downtown	Smith_Mike@yahoo.com
102	Smith	S.	Graham	2348	Checking	Bridgewater	Smith_Graham@rediffmail.com
103	Langer	G.	Justin	3421	Savings	Plainsboro	Langer_Justin@yahoo.com
104	Quails	D.	Jack	2367	Checking	Downtown	Quails_Jack@yahoo.com
105	Jones	E.	Simon	2389	Checking	Brighton	Jones_Simon@rediffmail.com

Customer_Detail records from Customer_Details table

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)
- Tuples are unique
- 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.

PROPERTIES OF RELATIONS

- There are no duplicate tuples
- Tuples are unordered,top to bottom
- Attributes are unordered , left to right
- All attribute values are atomic (I NF)

INTEGRITY CONSTRAINTS

1. Entity Integrity

   No component of a primary key can have NULL values

2. Referential Integrity

   Foreign Key column values are required to match values of candidate key in the same or another table

Keys

- IDENTIFICATION OF A PRIMARY KEY

STEPS:

1. IDENTIFY SUPER KEYS
2. IDENTIFY CANDIDATE KEYS OUT OF SUPER KEYS
3. SELECT PRIMARY KEY OUT OF CANDIDATE KEYS

- Super key
  - A combination of one or more attributes that uniquely identifies each row (entity instance).

Keys

- Candidate key
  - A Candidate key is a super key that does not have a subset which itself can uniquely identify each row.

Cust_ID	Cust_Last_ Name	Cust_Mid _Name	Cust_First _Name	Account _No	Account_ Type	Bank_Branch	Cust_Email
101	Smith	A.	Mike	1020	Savings	Downtown	Smith_Mike@yahoo.com
102	Smith	S.	Graham	2348	Checking	Bridgewater	Smith_Graham@rediffmail.com
103	Langer	G.	Justin	3421	Savings	Plainsboro	Langer_Justin@yahoo.com
104	Quails	D.	Jack	2367	Checking	Downtown	Quails_Jack@yahoo.com
105	Jones	E.	Simon	2389	Checking	Brighton	Jones_Simon@rediffmail.com

Customer_Detail records from Customer_Details table

- *Assumptions*
- *One Customer can have only one account*
- *An account can belong to only one Customer*

Keys

- 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 each row in the given table(primary key). Rest of the candidate keys are called Alternate Keys.
  - Give preference to numeric column(s)
  - Give preference to single attribute
  - Give preference to minimal composite key

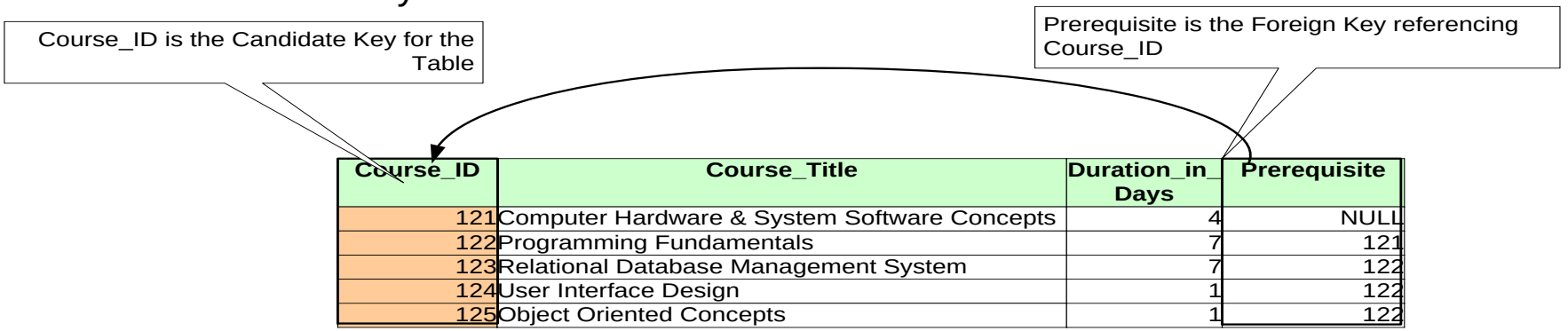
Keys

- 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 any of the Candidate keys..

Keys

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



Course_ID	Course_Title	Duration_in_Days	Prerequisite
121	Computer Hardware & System Software Concepts	4	NULL
122	Programming Fundamentals	7	121
123	Relational Database Management System	7	122
124	User Interface Design	1	122
125	Object Oriented Concepts	1	122

- Points to remember
  - 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

KEYS

Deptno is the primary key

DEPT

Deptno	Dname	Loc
10	EDP	Hyderabad
20	ACCOUNTS	Chennai
30	HRD	Bangalore

Deptno is the foreign key referencing primary key of Dept Table

Empno is the primary key

EMP

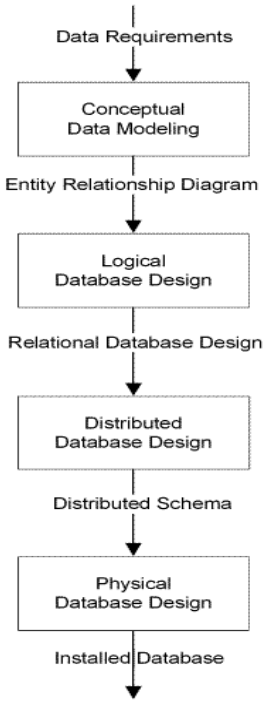
Empno	Ename	Job	Salary	Deptno
1011	Ravi Kumar	Manager	28000	10
1012	Laxmi	Programmer	8500	20
1013	Madhav	Analyst	12000	10

Data Modeling

Data Modeling

- **Entity relationship diagram (ERD)** is one of the most widely used technique for data modeling.
- Data modeling is an essential component of database design and development. It provides a means to analyze business requirements so as to standardize organizational vocabulary, enforce business rules, and ensure adequate data quality.

Data modeling is performed during the initial phases of the database development process (also referred as database life cycle) as shown in the following figure



During the conceptual data modeling phase, data requirements are expressed through an ERD. The conceptual data modeling phase in general is independent of a DBMS.

The logical design phase transforms the conceptual data model into a format understandable to DBMS. This phase may also enhance or refine the data model (ERD) of the previous phase to ensure efficient utilization of the database.

Since most of the commercial DBMS are based on the relational model, the end product of this phase is relational model design

Data Modeling

- The three levels of data modeling are **conceptual data model, logical data model, and physical data model.**

- The table below compares the different features:

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

Data Modeling

One of the ways an ERD is enhanced during the logical design phase is through the process of normalization.

It is the process of removing redundancy in a table so that the table is easier to modify.

It usually involves dividing an entity table into two or more tables and defining relationships between the tables. The objective is to isolate data so that additions, deletions, and modifications of an attribute can be made in just one table and then propagated through the rest of the database via the defined relationships.

- ***Top down Approach***
  - E R Modeling
- ***Bottom Up approach***
  - Normalization

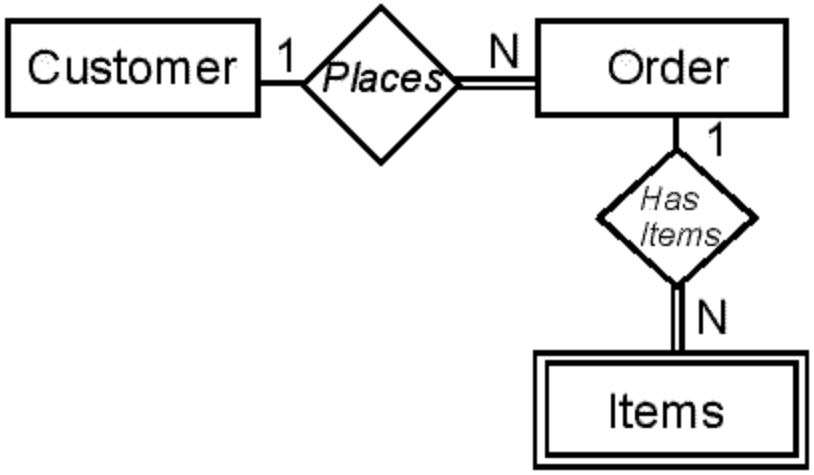
ER modeling

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

- **Entity**: **An entity is an object that exists and is distinguishable from other object.** Also known as **Entity type**. Ex. Dept,Emp,Student,Supplier etc.

- **Entity instance**: A record/row of a file/table Ex. Details of a student
- **Attributes**: Properties/characteristics that describe entities . Ex. Student_name,DOB ,student_code ex.

- **Relationships**: Associations between entities



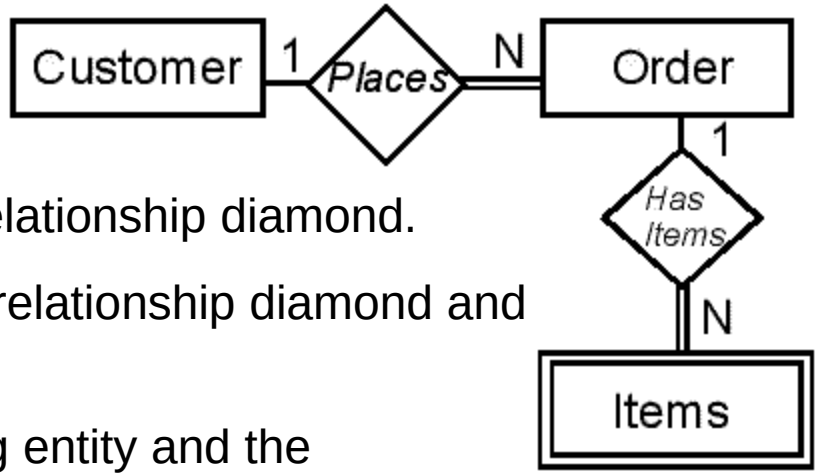
ER modeling

- **Relationship:** An association between two entities.
  A CUSTOMER *places* a CUSTOMER ORDER
  An EMPLOYEE *takes* a CUSTOMER ORDER
  A STUDENT *enrolls* in a COURSE
  A COURSE is *taught* by a FACULTY MEMBER

- Relationships are typically given names.
- A relationship can include one or more entities
- The *degree* of a relationship is the number of Entities that participate in the relationship.
- Relationships of degree 2 are called *binary relationships*. Most relationships in databases are binary.
- Relationship **Cardinality** refers to the number of entity instances involved in the relationship.
- For example:
  one CUSTOMER may place *many* CUSTOMER ORDERS
  many STUDENTS may sign up for *many* CLASSES
  one EMPLOYEE receives *one* PAYCHECK

ER modeling

- *Participation* of instances in a relationship may be mandatory or optional.
- For example,
  one CUSTOMER *may* place many CUSTOMER ORDERS
  one EMPLOYEE *must* fill out one or more PAY SHEETS
- This is also called "minimal cardinality" or the "optionality" of a relationship.



Chen notation

Relationship Name: Displayed just inside the relationship diamond.

Degree: Shown by line segments between the relationship diamond and 2 or more entities.

Cardinality: Displayed between the participating entity and the relationship diamond next to the relationship line. Split up the cardinality.

Optionality: Mandatory participation indicated by double relationship line
Optional participation indicated by a single relationship line.

Attributes

- The set of possible values for an attribute is called the **domain** (also called as **ValueSet**) of the attribute

  E.g.:
  - The domain of attribute ***marital status*** is just the four values: single, married, divorced, widowed
  - The domain of the attribute month is the twelve values ranging from January to December

- **Key attribute***:* The attribute (or combination of attributes) that is unique for every entity instance.The main purpose of a key attribute is to uniquely identify each row.

  E.g. the account number of an account, the employee id of an employee etc.

- If the key attribute consists of two or more attributes in combination, it is called a **composite key**

Simple Vs composite attribute

- **Simple attribute**: cannot be divided into simpler components

  E.g. Job of an employee

- **Composite attribute**: can be split into components

  E.g. Address of the employee.

  - » Can be split into door no, street no. , city etc.

Single Vs Multi-valued Attributes

- **Single valued** : can take on only a single value for each entity instance

  E.g. ***job*** of employee. There can be only one value for this

- **Multi-valued**: can take many values for each entity instance.
-

  E.g. ***skill set*** of employee

Stored Vs Derived attribute

- **Stored Attribute**: Attribute that needs to be stored permanently.

  E.g.: ***name*** of an employee

- **Derived Attribute**: Attribute that can be calculated based on other attributes

  E.g. : ***years of service*** of employee can be calculated from date of joining and current date

Regular Vs. Weak entity type

- **Regular Entity**: Entity that has its own key attribute(primary key).
- Regular Entity is also called as strong entity.

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

- **Weak entity**: Entity whose existence depends on other entity and doesn't have key attribute of its own

E.g. : spouse of employee

Relationships

- A **relationship type** between two entity types defines the set of all associations between these entity types

- Each instance of the relationship between members of these entity types is called a **relationship instance**

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** items from a 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**)

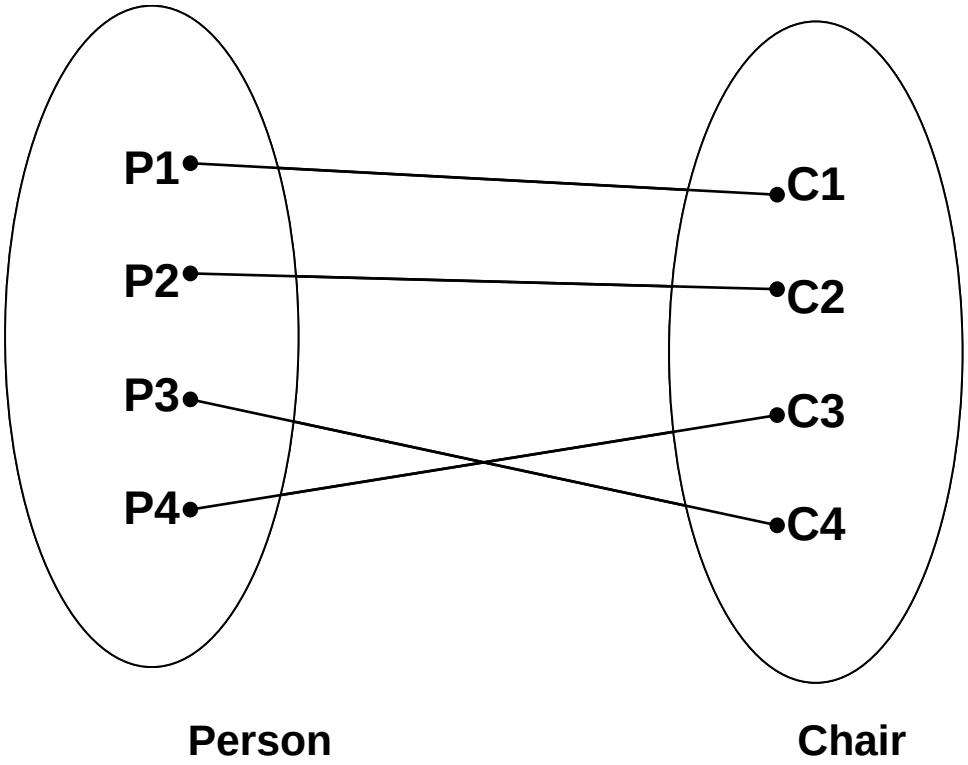
E.g.:

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

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

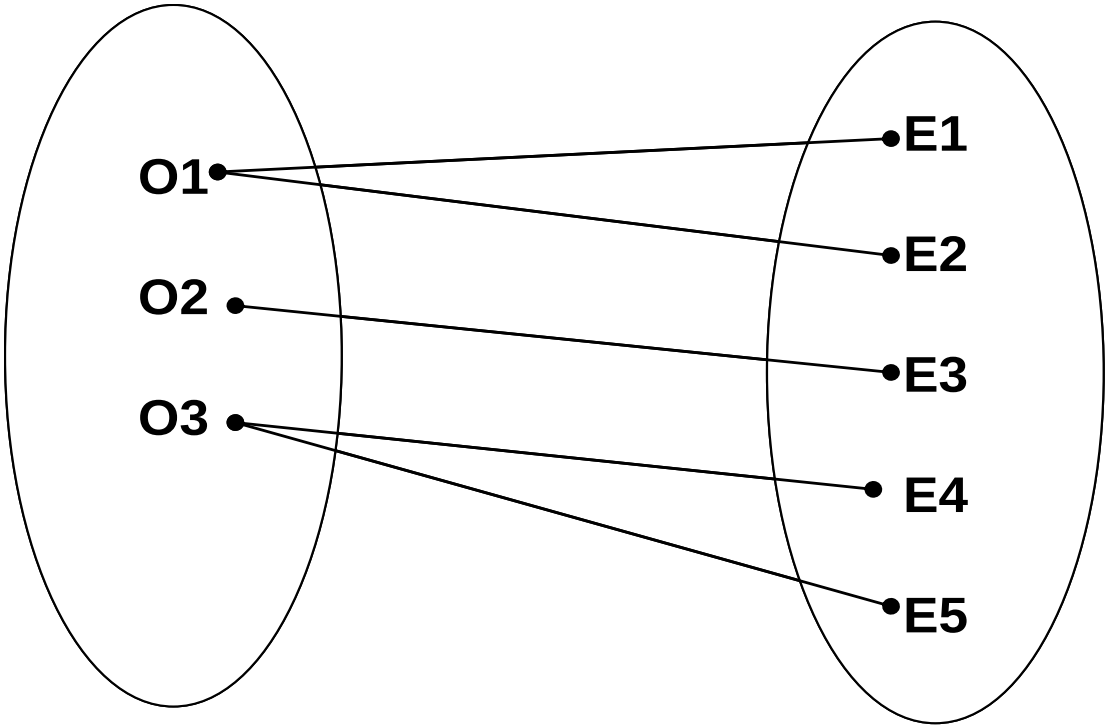
Student **enrolls** course (**M:N**)

Cardinality – One - To - One



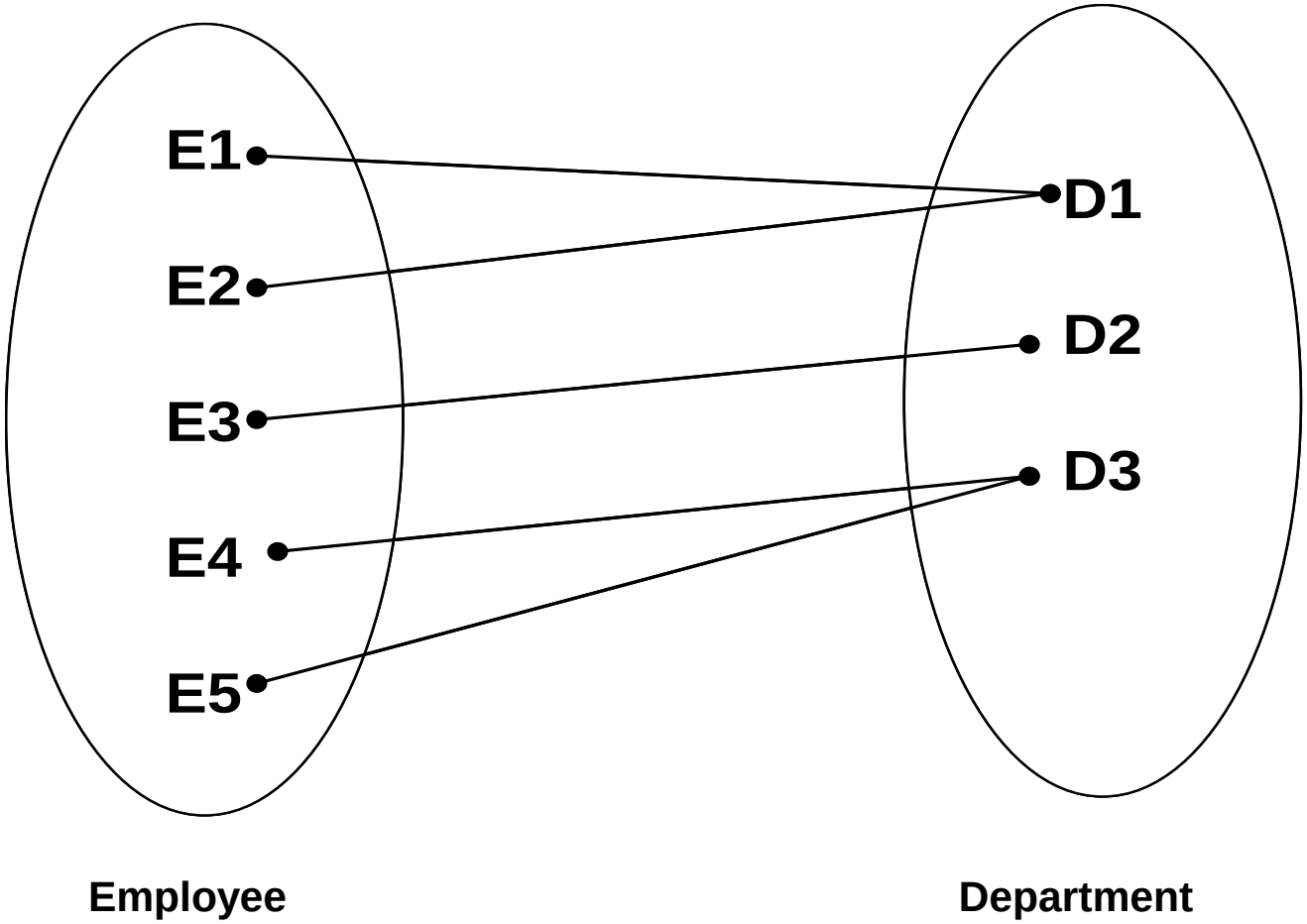
One instance of entity type Person is related to one instance of the entity type Chair.

Cardinality – One -to- Many



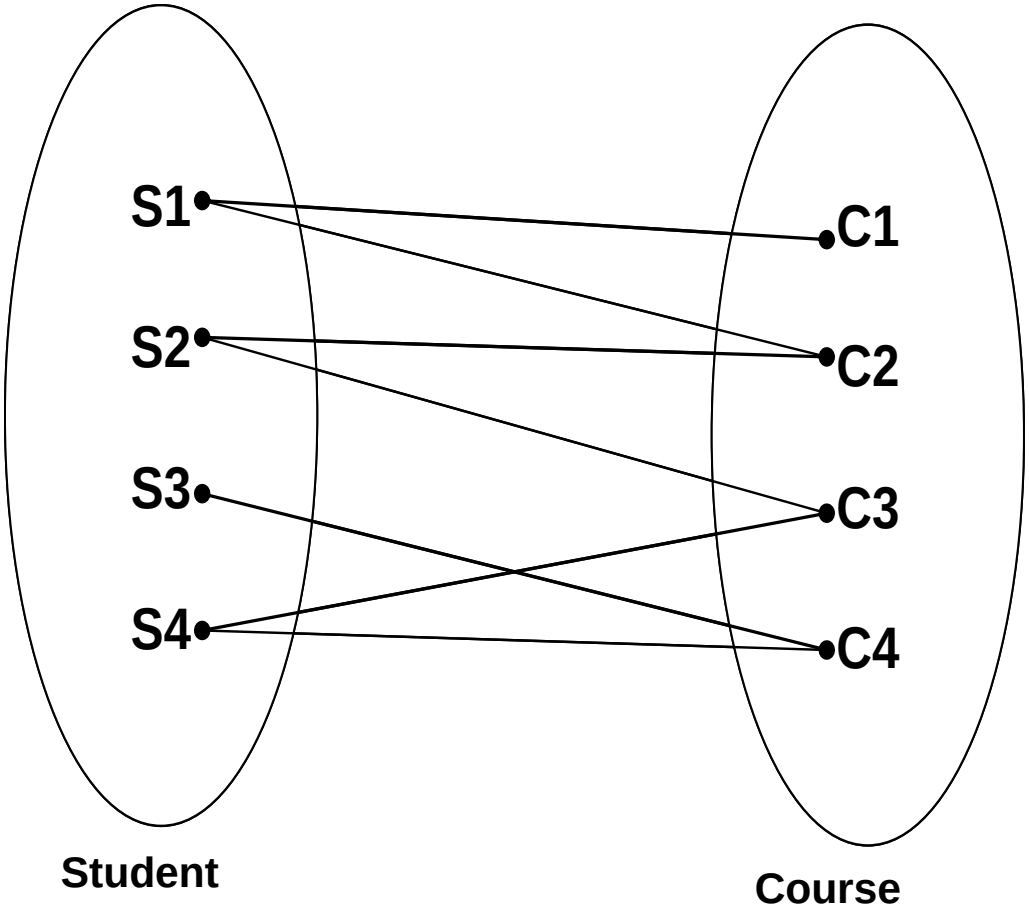
One instance of entity type Organization is related to multiple instances of entity type Employee

Cardinality – Many-to-One



Reverse of the One to Many relationship.

Cardinality – Many-to-Many



Multiple instances of one Entity are related to multiple instances of another Entity.

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
  Employee: partial
  Department: total

ER Modeling -Notations

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.

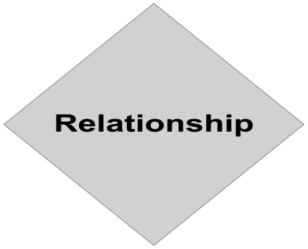
Attributes are the properties or characteristics of an Entity

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

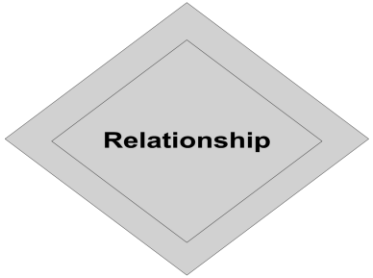
A multi-valued attribute can have more than one value. For example, an employee Entity can have multiple skill values.

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.

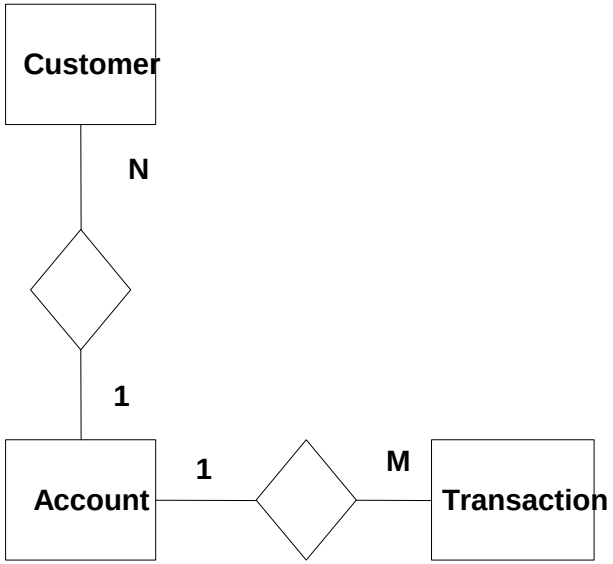


Relationships illustrate how two entities share information in the database structure.

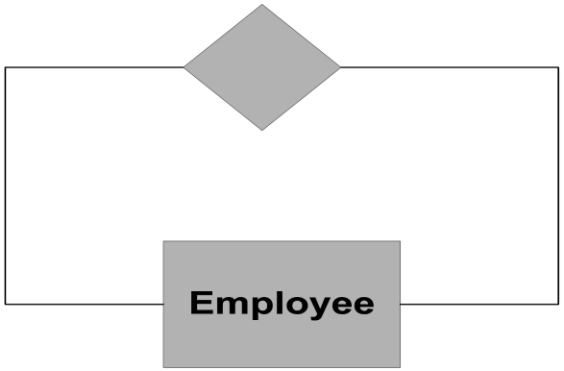


To connect a weak Entity with others, you should use a weak relationship notation.

ER Modeling -Notations

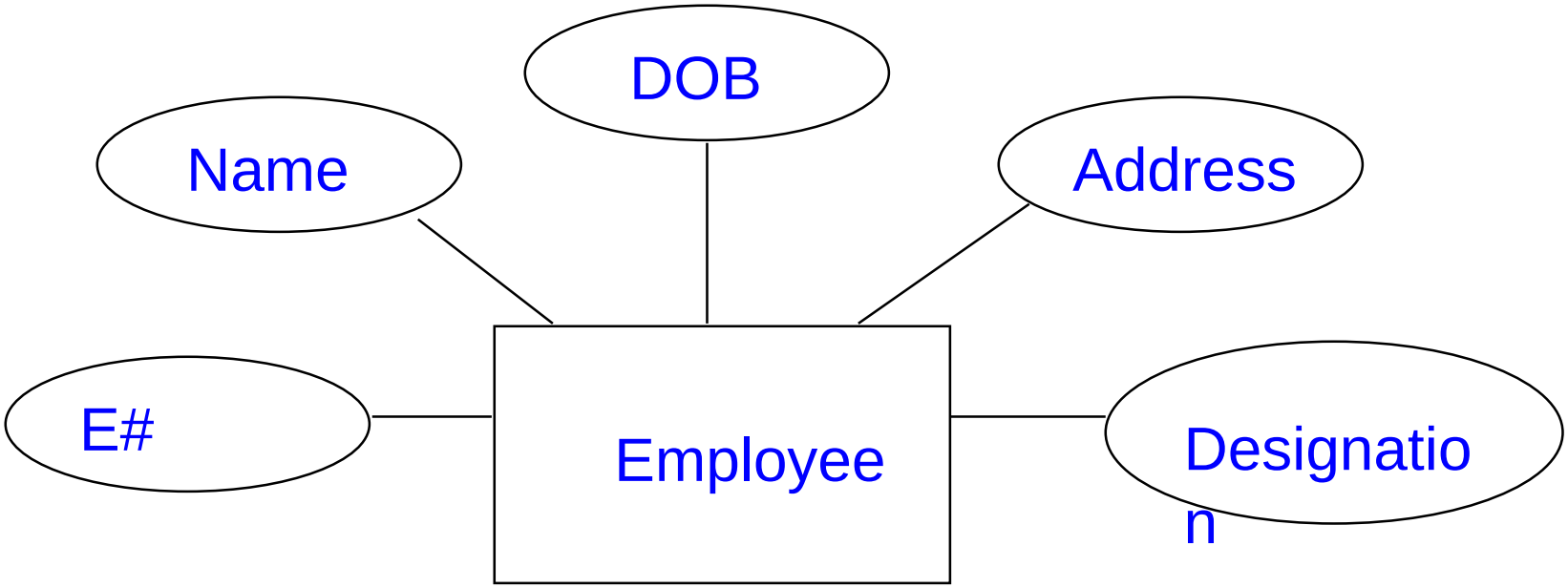


Cardinality specifies how many instances of an Entity relate to one instance of another Entity. M,N both represent 'MANY' and 1 represents 'ONE' Cardinality

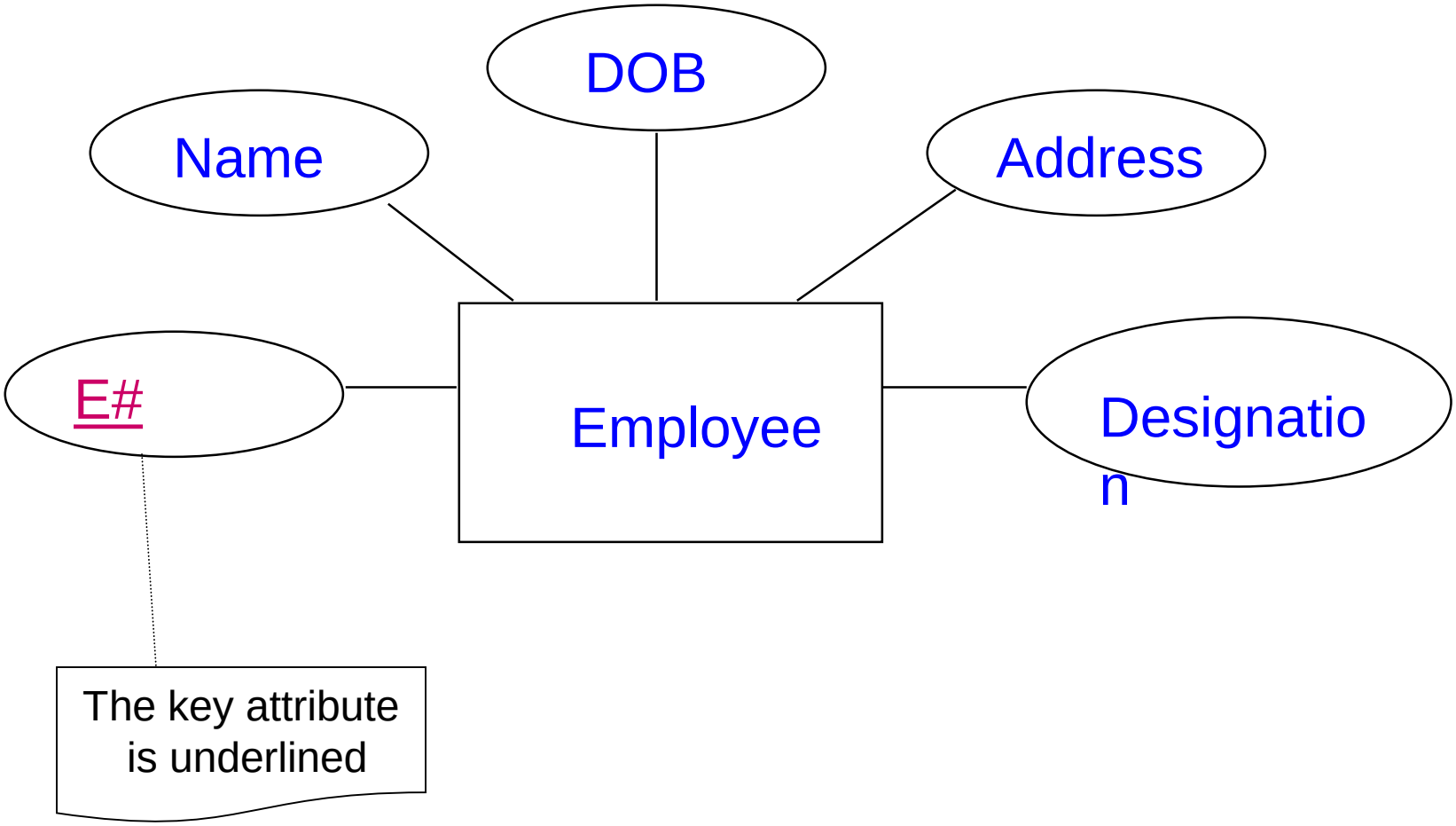


In some cases, entities can be self-linked. For example, employees can supervise other employees

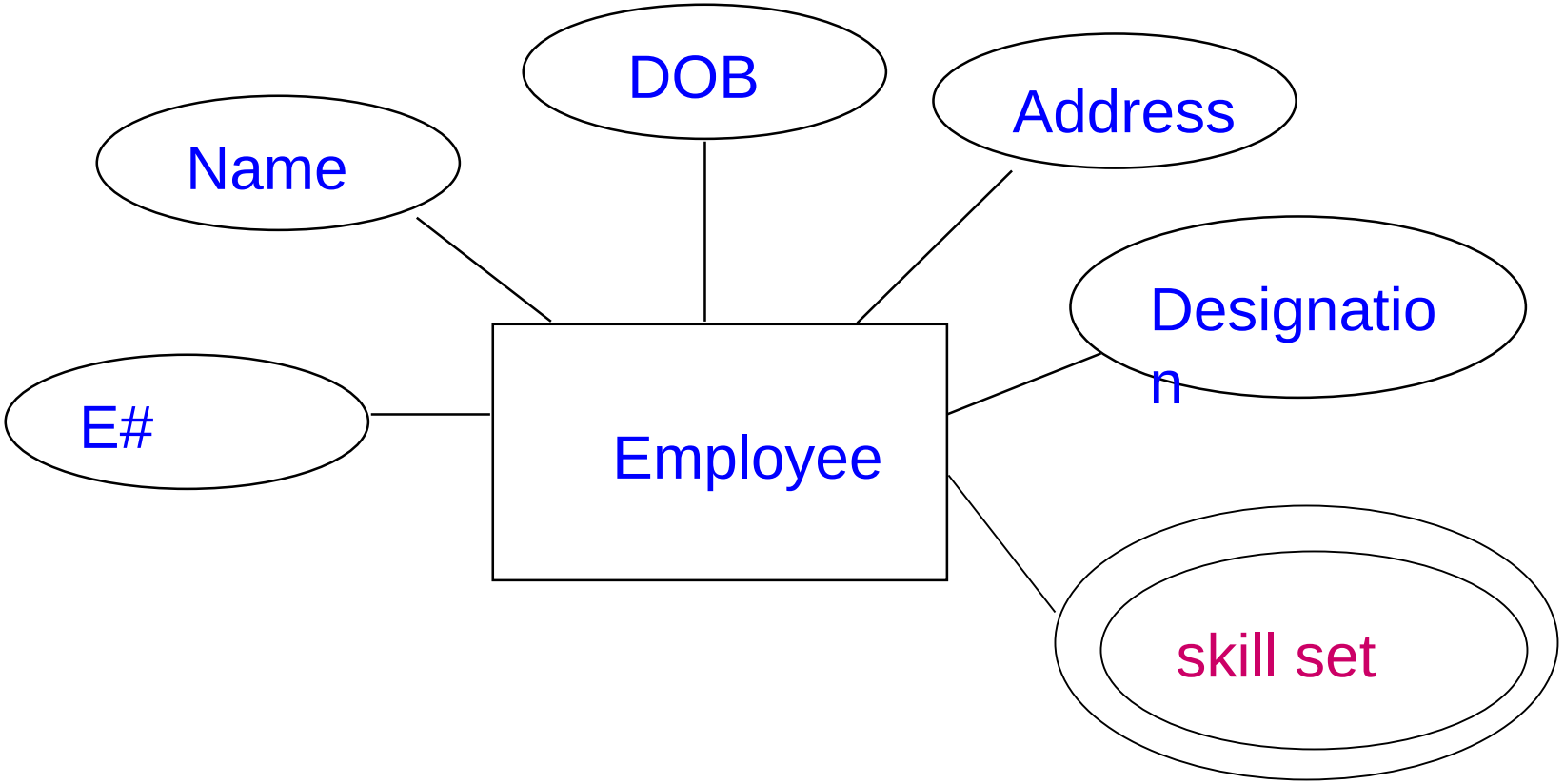
Attributes



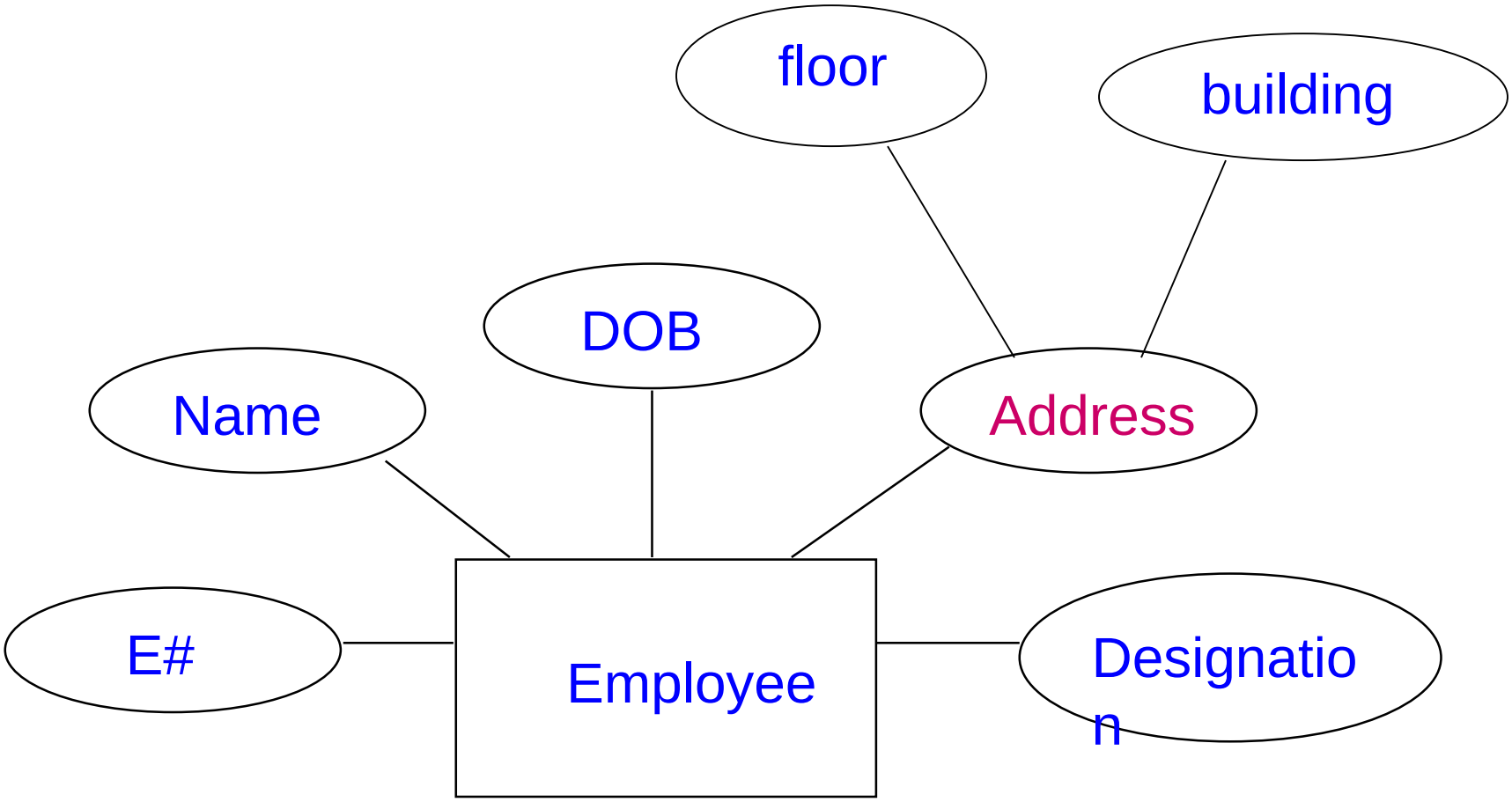
Key attribute



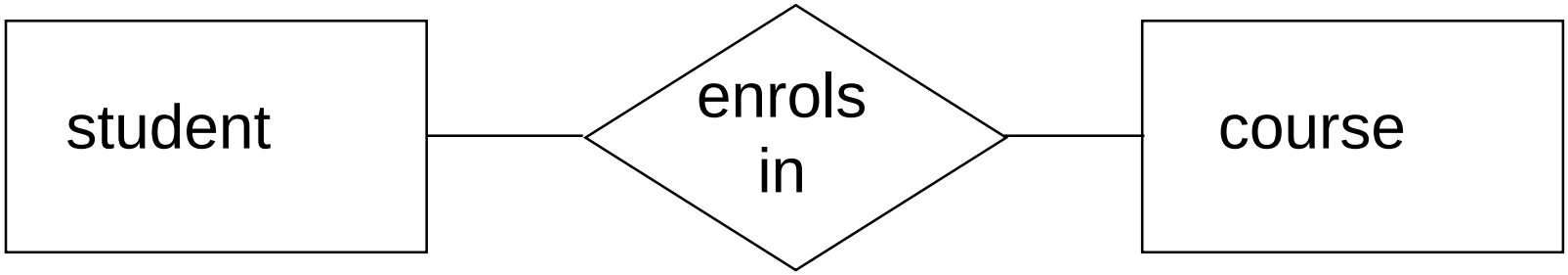
Multivalued Attribute



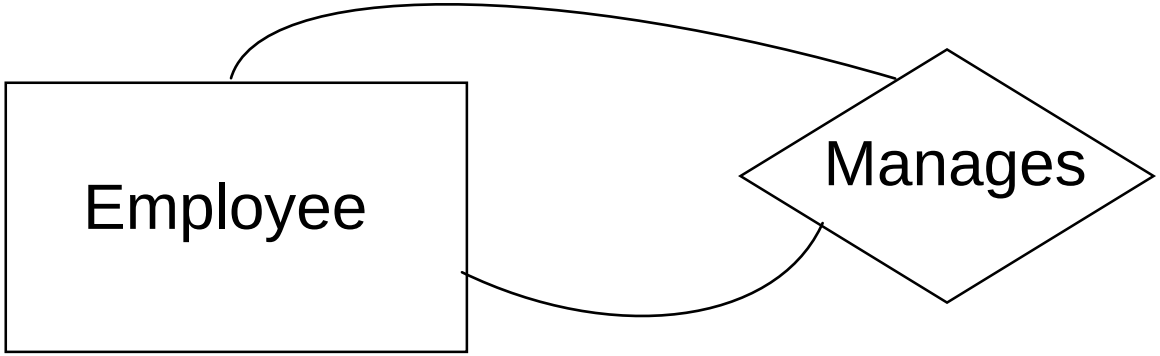
Composite attribute



Relationship

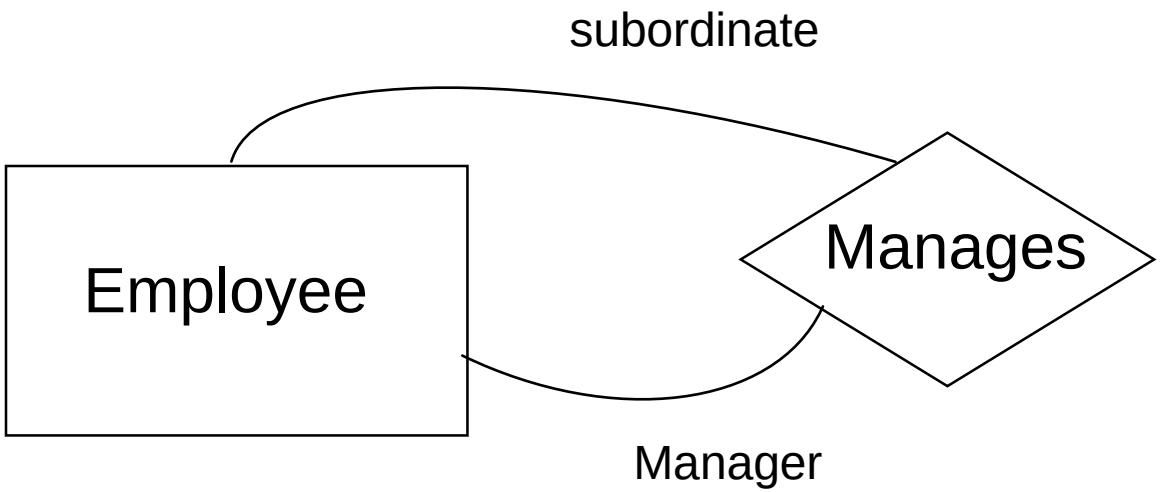


Unary Relationship

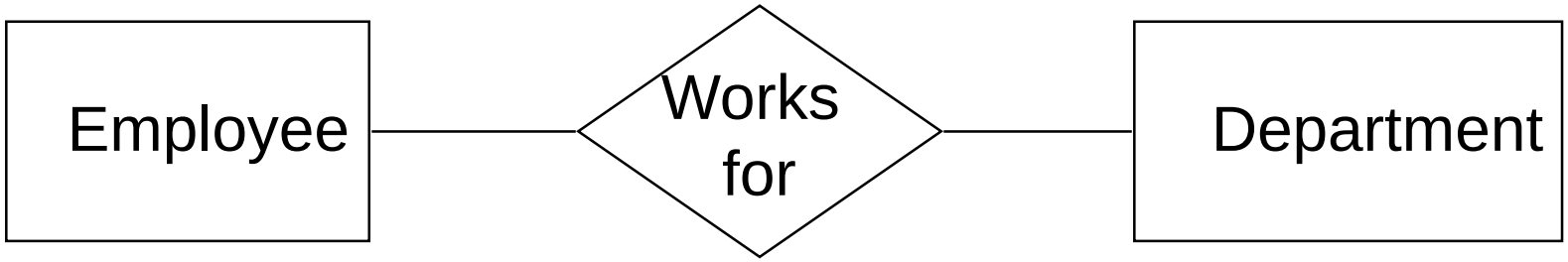


Role names

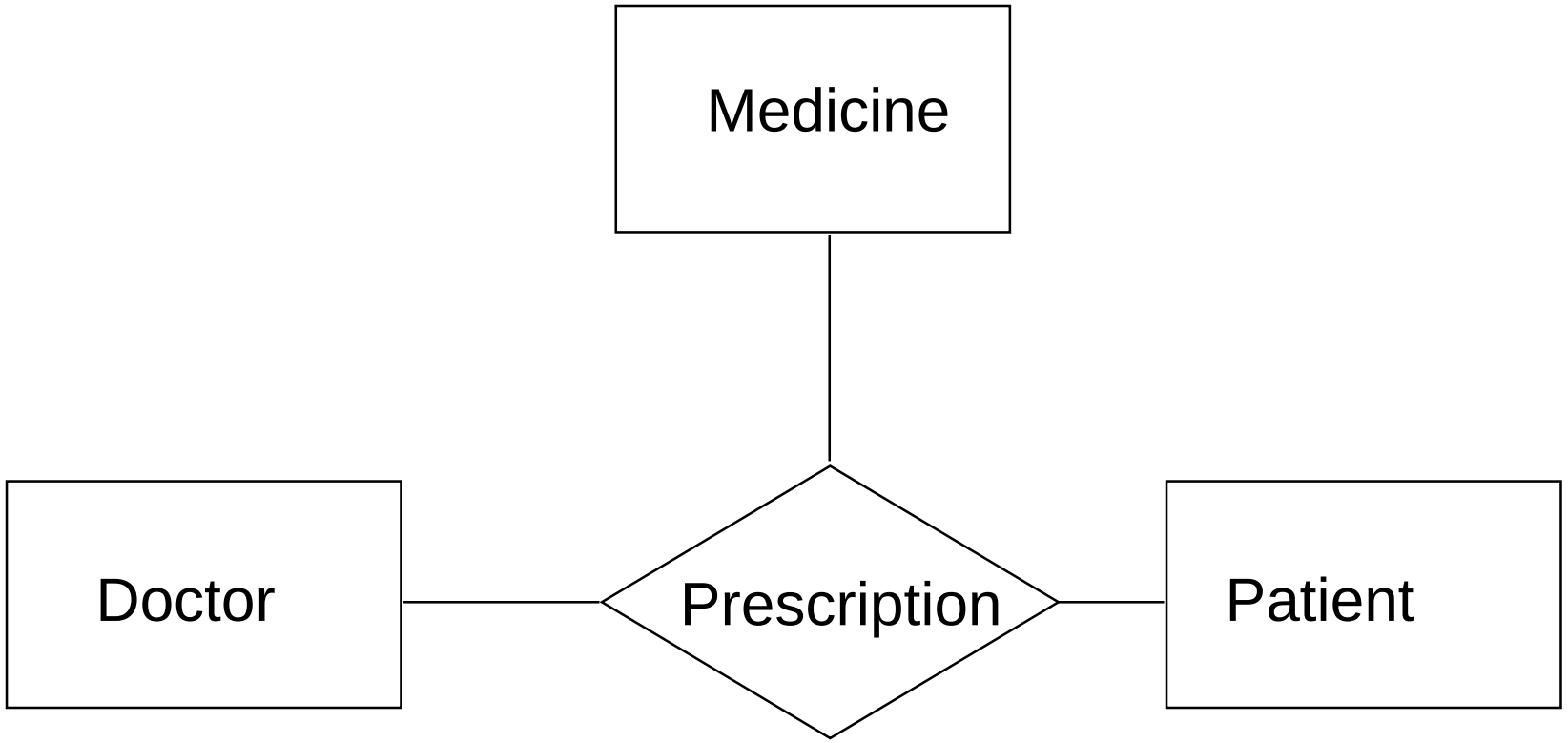
- Role names may be added to make the meaning more explicit



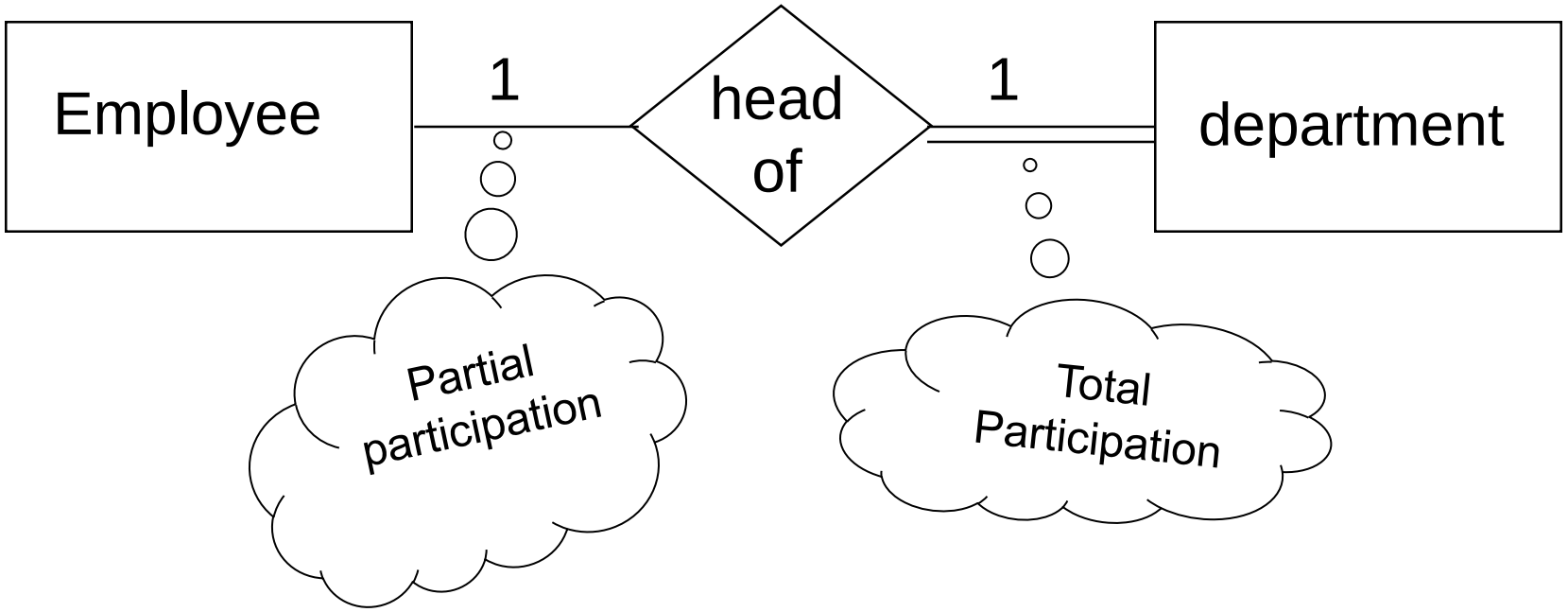
Binary Relationship



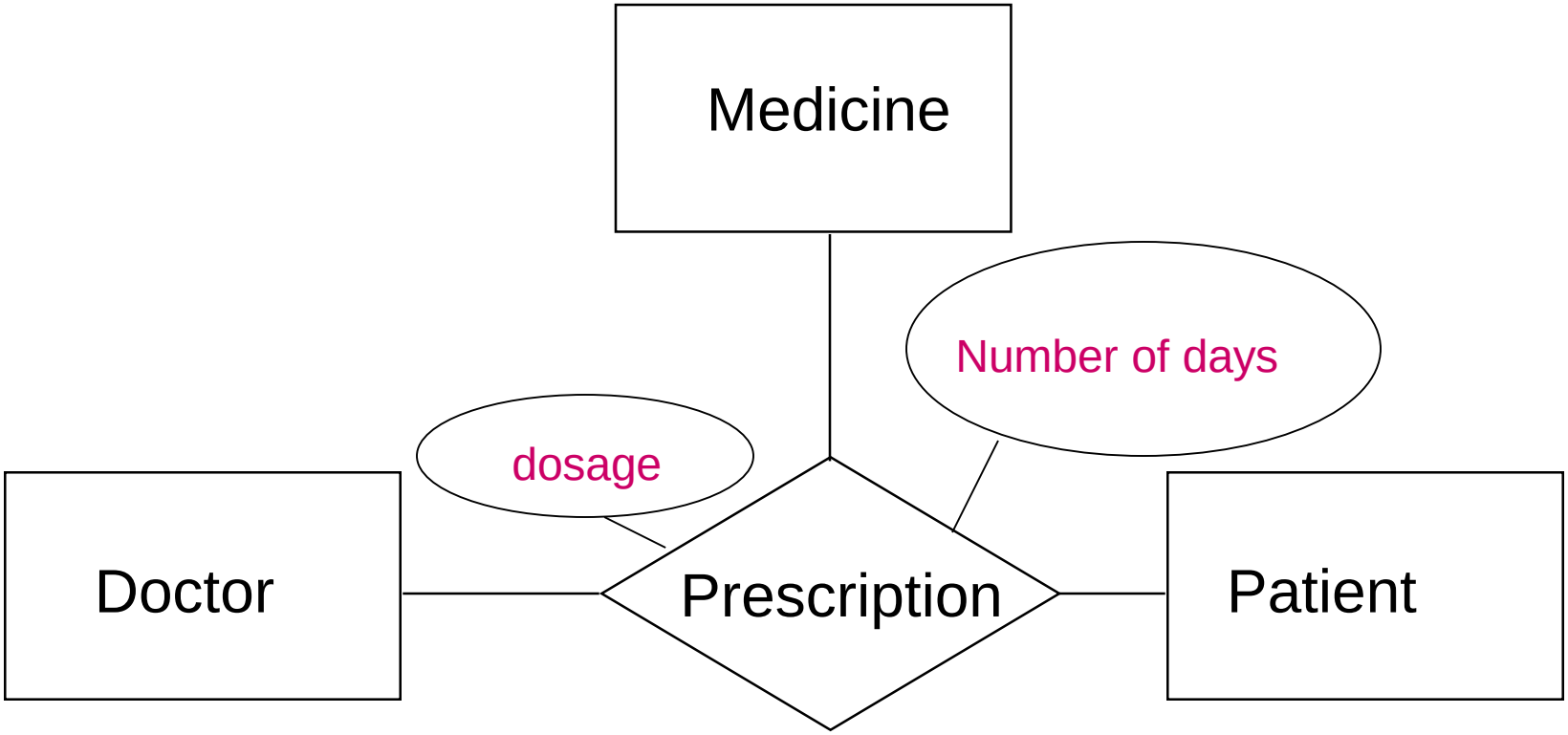
Ternary Relationship



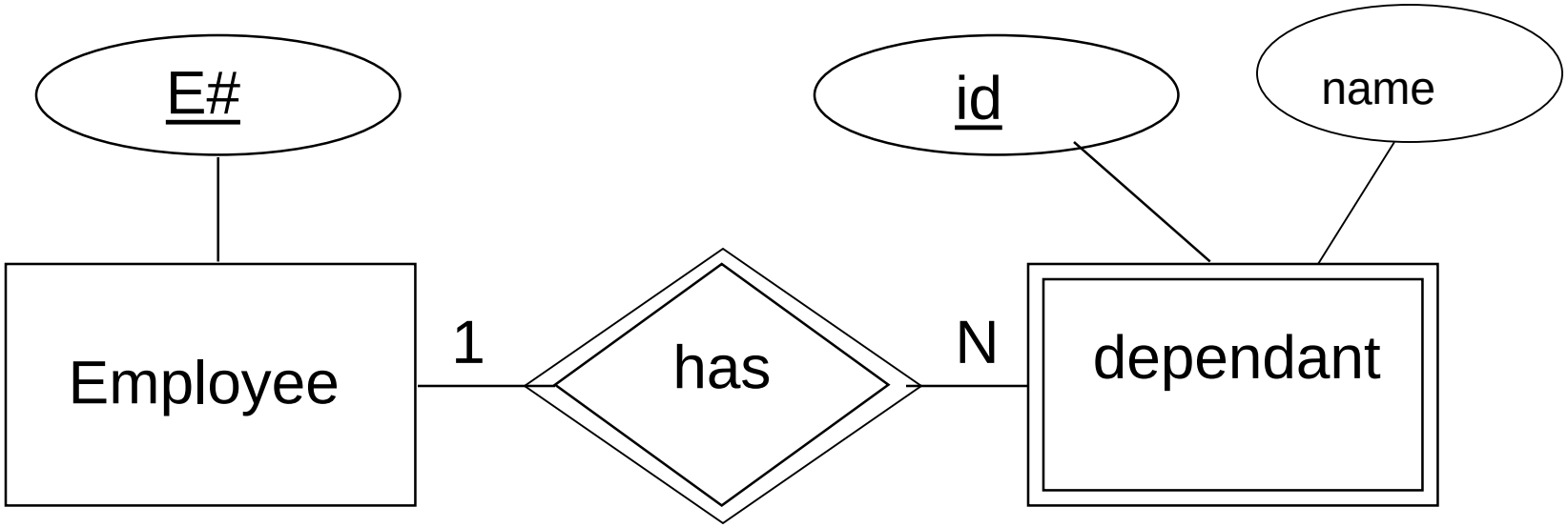
Relationship participation



Attributes of a Relationship



Weak entity



The dependant entity is represented by a double lined rectangle and the identifying relationship by a double lined diamond

Case Study – ER Model For a college DB

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

Steps in ER Modeling

- Identify the Entities
- Find the 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

Step 1 : Identify the Entities :

- **DEPARTMENT**
- **STUDENT**
- **COURSE**
- **INSTRUCTOR**

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**.

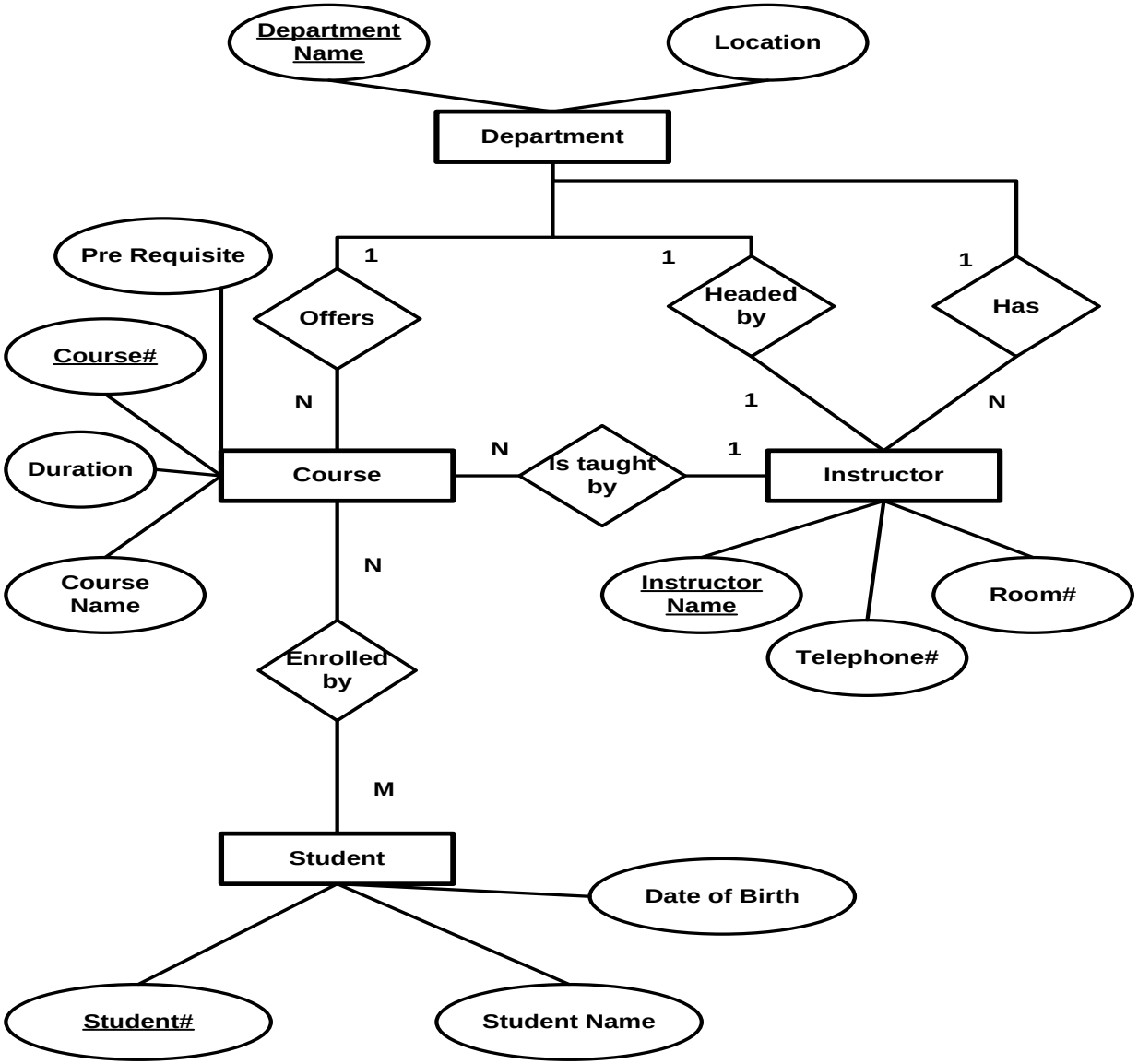
Step 3: Identify the key attributes

- **Deptno** 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#** is the key attribute for “Instructor” Entity.

Step 4: Identify other relevant attributes

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

Step 5: Draw complete E-R diagram with all attributes including Primary Key



Case Study – Banking Business Scenario

Assumptions :

- 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

Steps in ER Modeling

- Identify the Entities
- Find the 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
- Create relations

Step 1 : Identify the Entities :

- **BANK**
- **BRANCH**
- **LOAN**
- **ACCOUNT**
- **CUSTOMER**

Step 2 : Find the relationships

- 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**.

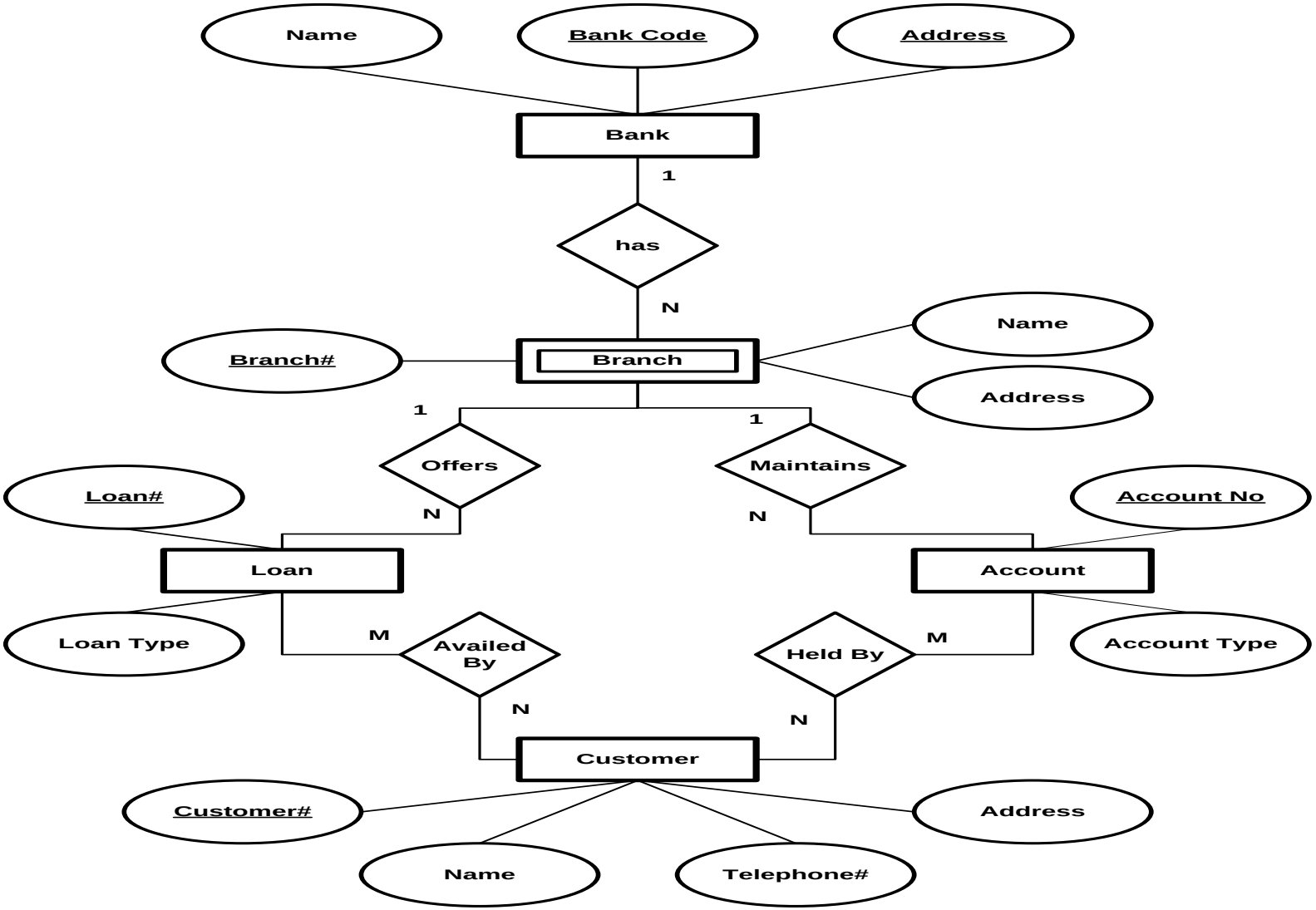
Step 3: Identify the key attributes

- **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: Identify other relevant attributes

- For the “Bank” Entity, the relevant attributes other than “Bank Code” would be “Name” and “Address”.
- For the “Branch” Entity, the relevant attributes other than “Branch#” would be “Name” and “Address”.
- For the “Loan” Entity, the relevant attribute other than “Loan#” would be “Loan Type”.
- For the “Account” Entity, the relevant attribute other than “Account No” would be “Account Type”.
- For the “Customer” Entity, the relevant attributes other than “Customer#” would be “Name”, “Telephone#” and “Address”.

Step 5: Draw complete E-R diagram with all attributes including Primary Key



Merits and Demerits of ER Modeling

Merits

- Easy to understand. Represented in Business Users Language. Can be understood by non-technical specialist.
- *Intuitive* and helps in Physical Database creation.
- Can be generalized and specialized based on needs.
- Can help in database design.
- Gives a higher level description of the system.

Demerits

- Physical design derived from E-R Model may have some amount of ambiguities or inconsistency.
- Sometime diagrams may lead to misinterpretations

Logical database design

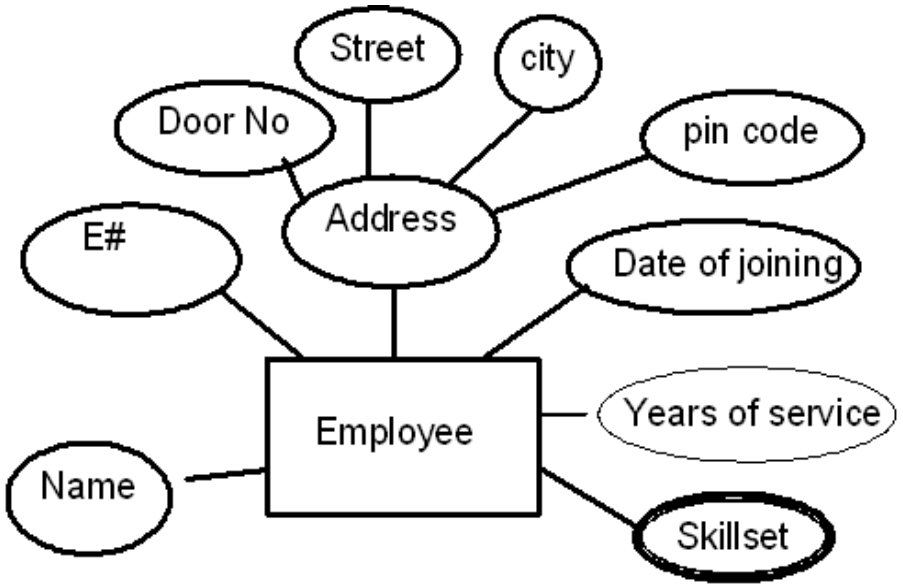
Converting ER diagrams to relational schema

Converting Strong entity types

- Each **entity type** becomes a **table**
- Each **single-valued** attribute becomes a **column**
- **Derived** attributes are **ignored**
- **Composite** attributes are represented by **components**
- **Multi-valued** attributes are represented by a **separate table**
- The **key** attribute of the entiry type becomes the **primary key** of the table

Entity example

- Here address is a composite attribute
- Years of service is a derived attribute (can be calculated from date of joining and current date)
- Skill set is a multi-valued attribute



- **The relational Schema**

Employee (E#, Name, Door_No, Street, City, Pincode, Date_Of_Joining)

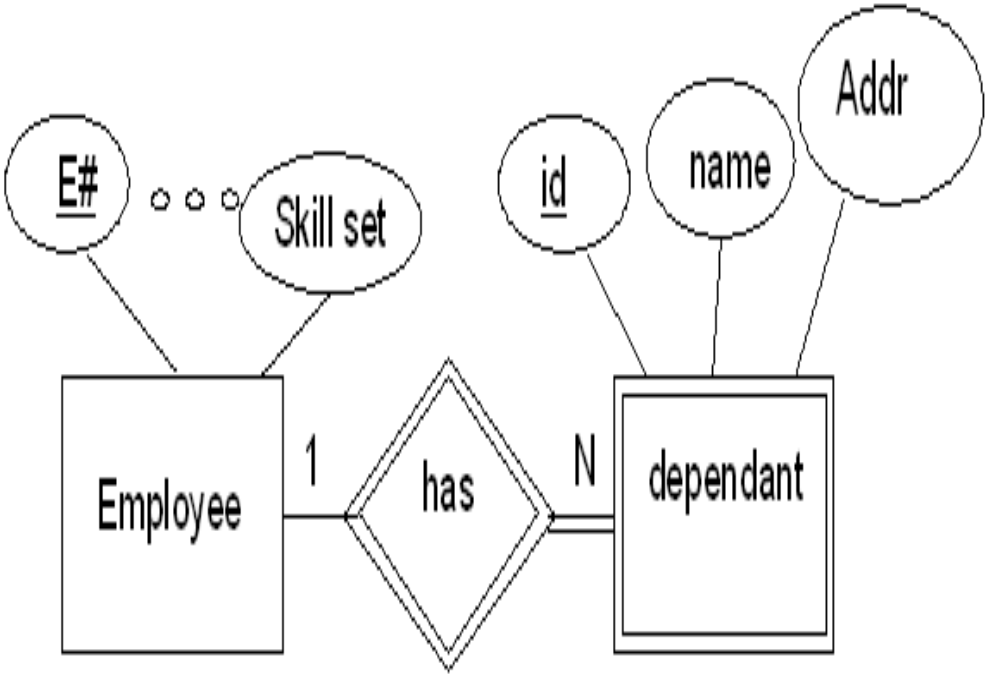
Emp_Skillset(E#, Skillset)

Converting weak entity types

- Weak entity types 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

- **The Relational Schema**



Employee (E# ,…….)

Dependant (E# , Dependant_ID, Name, Address)

Converting relationships

- The way relationships are represented depends on the cardinality and the degree of the relationship

- The possible cardinalities are:

  1:1, 1:M, N:M

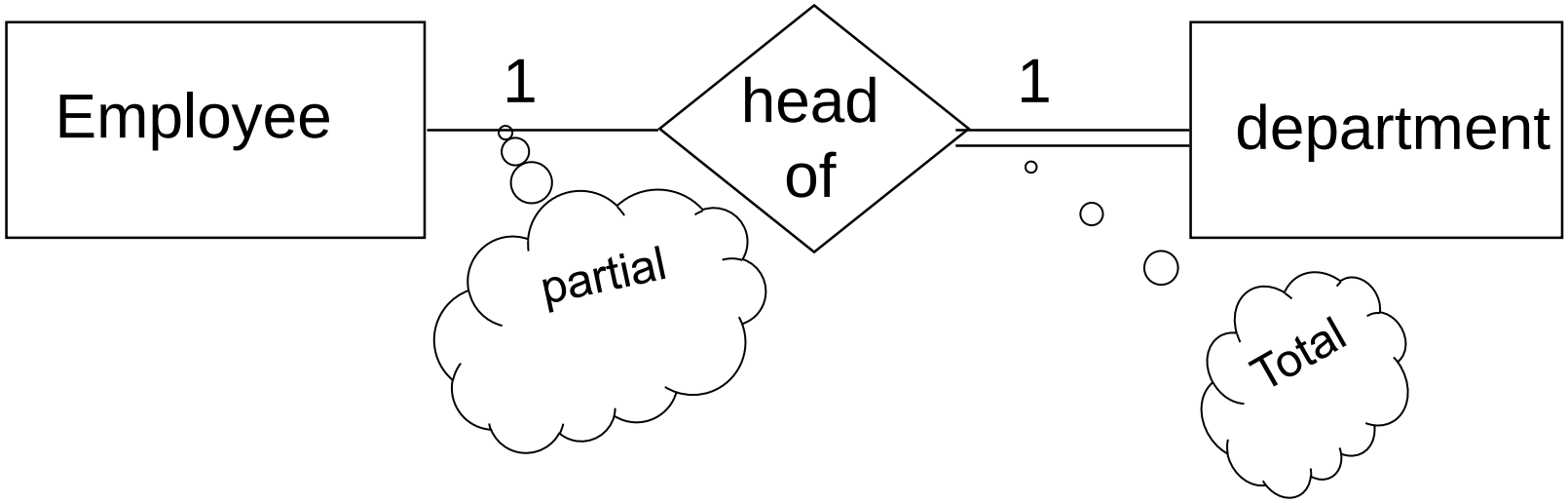
- The degrees are:

  Unary

  Binary

  Ternary …

Binary 1:1



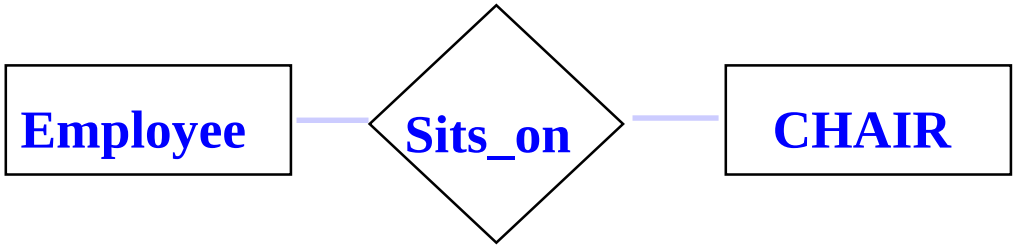
- Case 1:Combination of participation types

  The primary key of the partial participant will become the foreign key of the total participant

  Employee(E#, Name,…)

  Department (Dept#, Name…,Head)

Binary 1:1



- Case 2: Uniform participation types

  The primary key of either of the participants can become a foreign key in the other

  Employee (E#,name…)

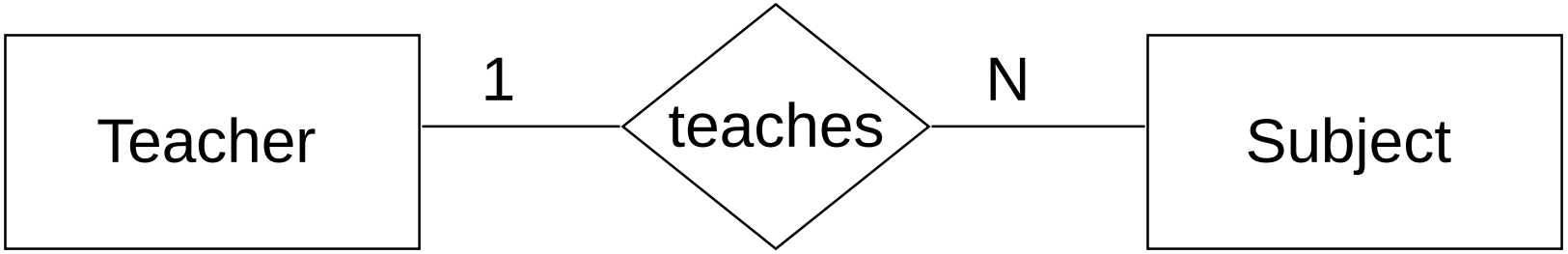
  Chair(item#, model, location, used_by)

  (or)

  Employee (E#, Name….Sits_on)

  Chair (item#,….)

Binary 1:N

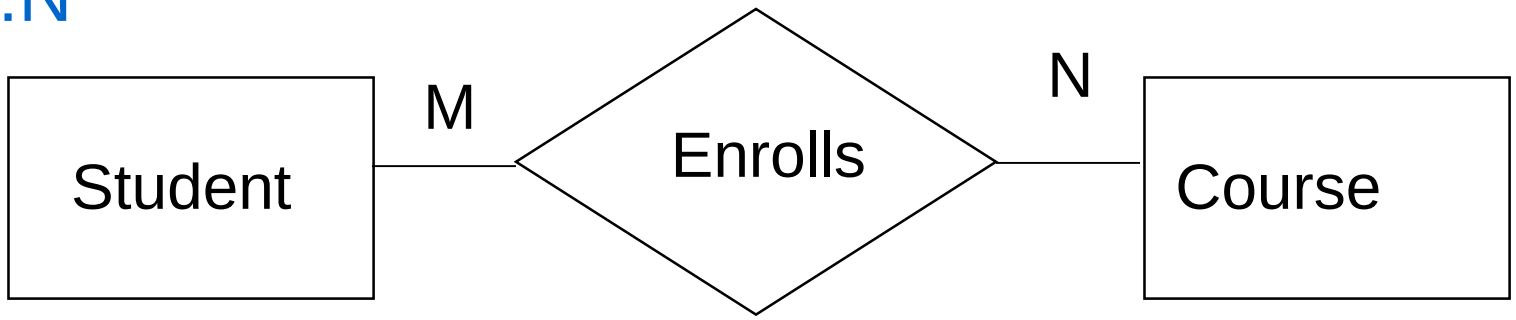


The primary key of the relation on the “1” side of the relationship becomes a foreign key in the relation on the “N” side

Teacher (<u>ID</u>, Name, Telephone, ...)

Subject (<u>Code</u>, Name, ..., Teacher)

Binary M:N



- A new table is created to represent the relationship
- Contains two foreign keys - one from each of the participants in the relationship
- The primary key of the new table is the combination of the two foreign keys

Student (Sid#,Title…) Course(C#,CName,…)

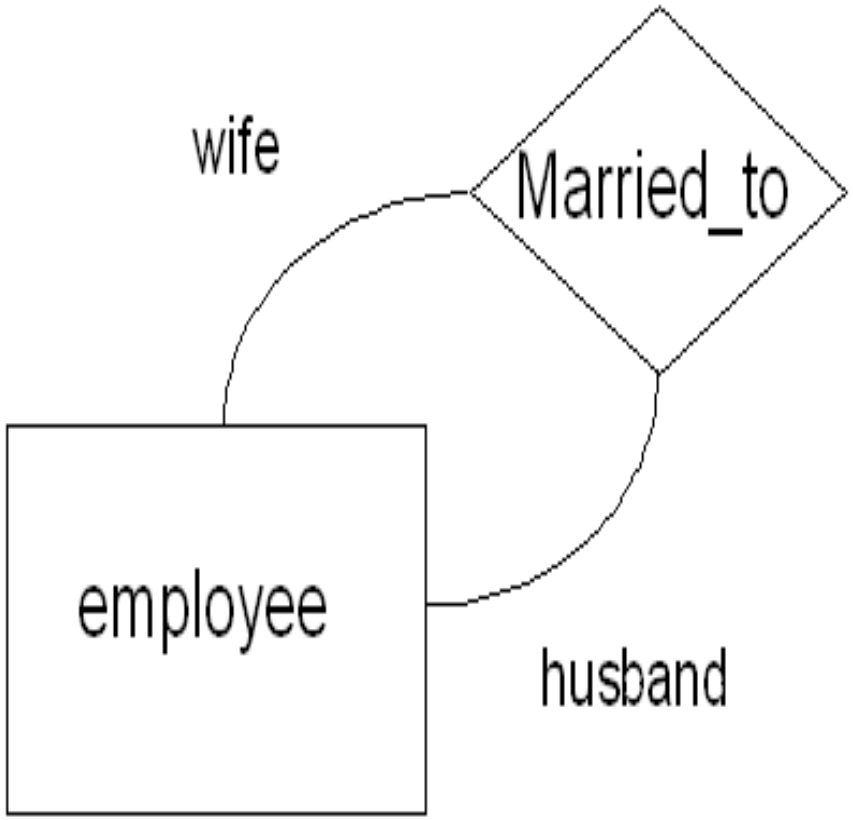
Enrolls (Sid#, C#)

Unary 1:1

• Consider employees who are also married couple

• The primary key field itself will become foreign key in the same table

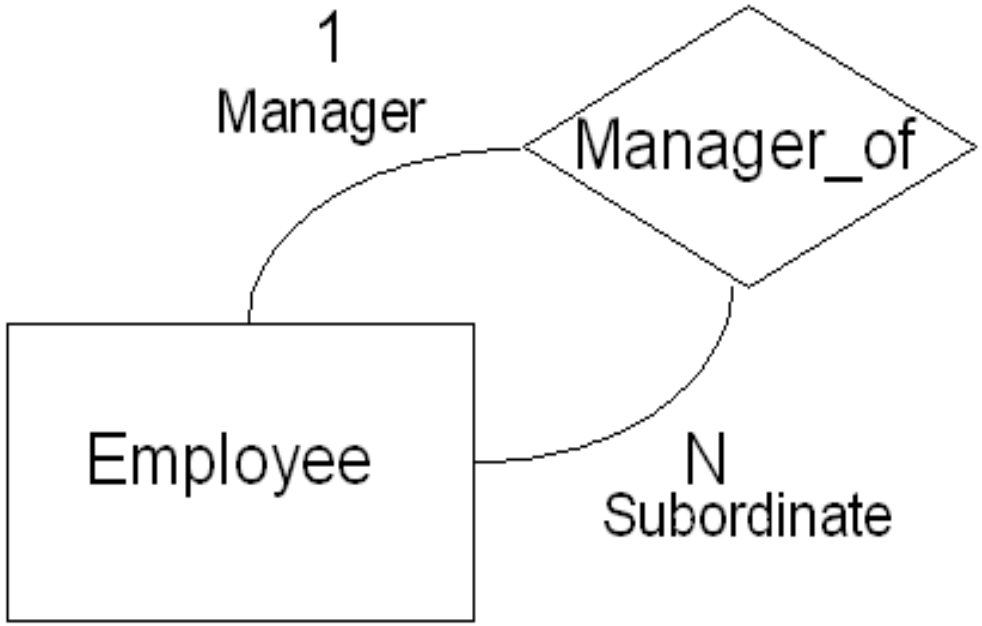
Employee(E#, Name,... Married_to)



Unary 1:N

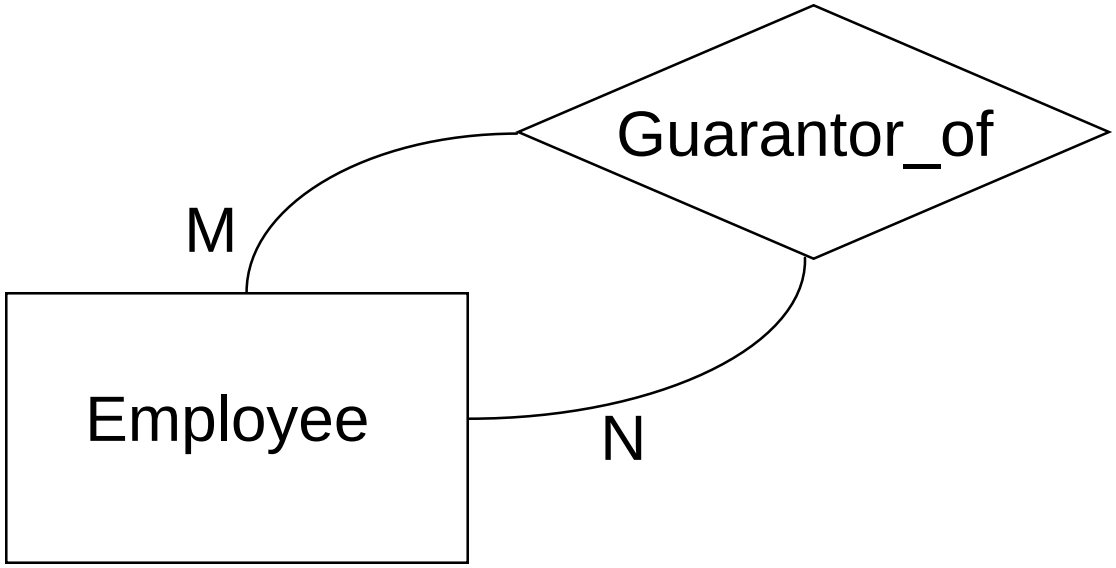
- The primary key field itself will become foreign key in the same table

- Same as unary 1:1



Employee(E#, Name,…,Manager)

Unary M:N



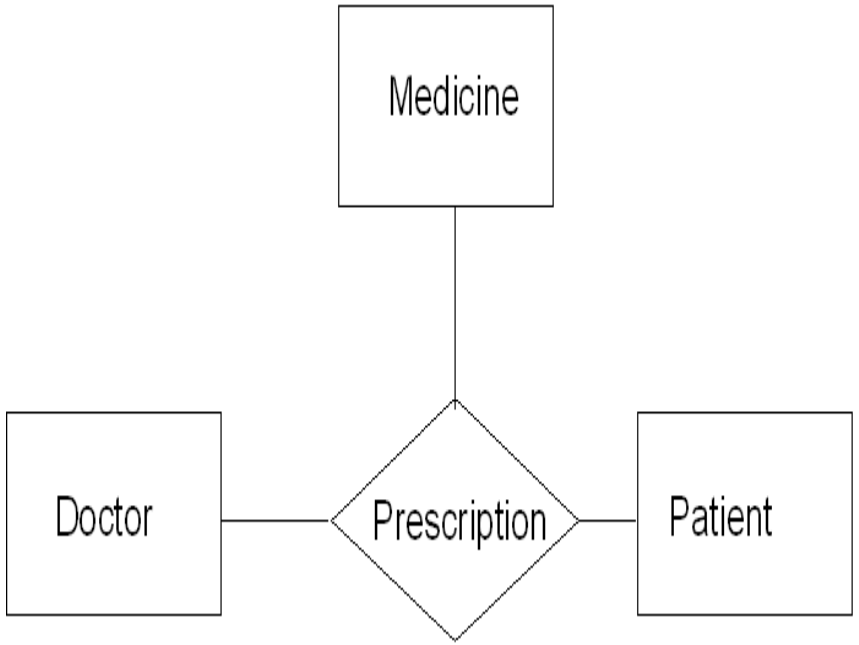
- There will be two resulting tables. One to represent the entity and another to represent the M:N relationship as follows

Employee(E#, Name,…)

Guaranty(Guarantor, beneficiary)

Ternary relationship

- Represented by a new table
- The new table contains three foreign keys - one from each of the participating Entities
- The primary key of the new table is the combination of all three foreign keys
- Prescription (Doctor#, Patient #, Medicine_Name)



Deriving Logical Schema for Banking Application

- Each Entity represented in the E-R model can be defined as a table in the relational scheme. All the attributes of the Entity will become columns of the table.

  - ***Example:*** *Let us consider the CUSTOMER Entity of the banking database scenario. We can translate this Entity to a "CUSTOMER" table with the following columns.*

- **CUSTOMER**
  - (Customer#, Name, Telephone#, Address)

  - ***Example:*** *Similarly a* ***"Bank"*** *table can be created with Bank Bankcode, Name and Address columns*
- **BANK**
  - (BankCode, Name, Address)

Deriving Logical Schema for Banking Application

- Weak Entity types 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.

- ***Example:*** *As per this guideline, a "Branch" table can be created with the following structure.*

- **BRANCH**
  - (BankCode, Branch#, Name, Address)

Deriving Logical Schema for Banking Application

- Each relationship can be defined as separate table in relational schema. Key attributes of *participating entities* will become key attribute of the Relationship.

  - ***Example:*** *We can define Loan_Detail table with Loan# and Customer# together as primary key with other relevant attributes like DateOfSanction, InterestRate, LoanAmount, Duration etc.*

- **LOAN_DETAILS**
  - (Loan#, Customer#, DateofSanction, InterestRate, LoanAmount, Duration)

Participating entities: The entities which are joined by the relation.

Deriving Logical Schema for Banking Application

- In a Many to Many relationship, it is necessary to create separate tables for participating entities and relationships.

- In the banking application, Customer and Loan Entities have a Many to Many relationship. Hence one should create separate tables for CUSTOMER, LOANS and LOAN_DETAILS. Here LOAN_DETAILS refers to relationship table.

Normalization

What is Normalization?

- Database designed based on the E-R model may have some amount of
  - Inconsistency
  - Uncertainty
  - Redundancy

To eliminate these draw backs some **refinement** has to be done on the database.

- **Refinement** process is called **Normalization**
- Defined as a step-by-step process of decomposing a complex relation into a simple and stable data structure.
- The 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.

Need for Normalization

Student_Course_Result Table

Student_Details			Course_Details				Result_Details		
101	Davis	11/4/1986	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	82	A
102	Daniel	11/6/1987	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	62	C
101	Davis	11/4/1986	H6	American History		4	11/22/2004	79	B
103	Sandra	10/2/1988	C3	Bio Chemistry	Basic Chemistry	11	11/16/2004	65	B
104	Evelyn	2/22/1986	B3	Botany		8	11/26/2004	77	B
102	Daniel	11/6/1987	P3	Nuclear Physics	Basic Physics	13	11/12/2004	68	B
105	Susan	8/31/1985	P3	Nuclear Physics	Basic Physics	13	11/12/2004	89	A
103	Sandra	10/2/1988	B4	Zoology		5	11/27/2004	54	D
105	Susan	8/31/1985	H6	American History		4	11/22/2004	87	A
104	Evelyn	2/22/1986	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	65	B

- Insert Anomaly
- Delete Anomaly
- Update Anomaly
- Data Duplication

Functional dependency

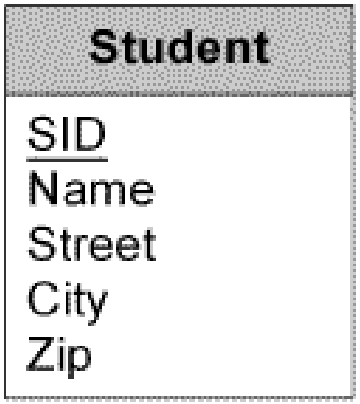
- In a given relation R, X and Y are attributes.
- Attribute Y is **functionally dependent** on attribute X **if each value of X determines EXACTLY ONE value of Y,** which is represented as X -> Y (X can be composite in nature).
- We say here “x determines y” *or* “y is functionally dependent on x”
  $X \rightarrow Y$ does <u>not</u> imply $Y \rightarrow X$
- If the value of an attribute “Marks” is known then the value of an attribute “Grade” is determined since Marks$\rightarrow$ Grade
- Types of functional dependencies:
  - Full Functional dependency
  - Partial Functional dependency
  - Transitive dependency

Functional dependency

- *Functional dependency* in an entity type occurs if one observes the association among the entity identifier and other attributes as reflected in an entity instance.
- Each entity instance represents a set of values taken by the non entity identifier attributes for each primary key (entity identifier) value.

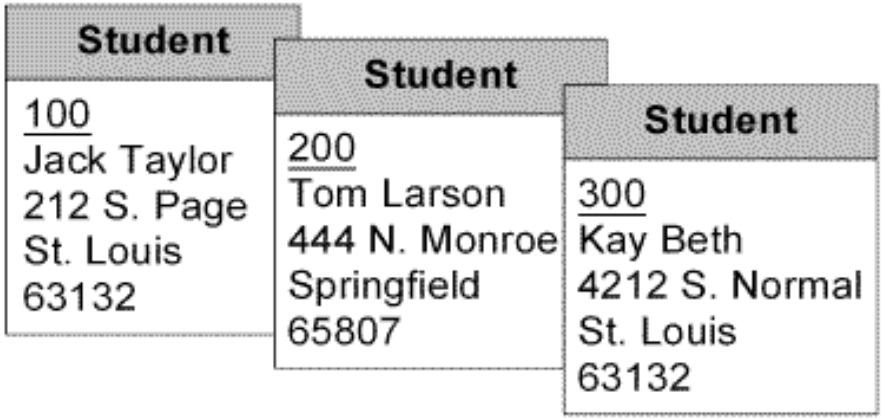
- So, in a way an entity instance structure also reflects an application of the functional dependency concept.

For example, the Student entity type of the following figure can represent the functional dependency **SID - > Name, Street, City, Zip.**



Functional dependency

- Each entity instance will now represent the functional dependency among the entity attributes as shown below.



- During requirement analysis, some entity types may be identified through functional dependencies, while others may be determined through database relationships. For example, the statement, "A faculty teaches many offerings but an offering is taught by one faculty" defines entity type Faculty and Offerings.
- Another important consideration is to distinguish when one attribute alone is the entity identifier versus a composite entity identifier. A composite entity identifier is an entity identifier with more than one attribute.
- ***A functional dependency in which the determinant contains more than one attribute usually represents a many-to-many relationship,*** which is more addressed through higher normal forms.

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

The first normal form rule is that there should be no nesting or repeating groups in a table.

Now an entity type that contains only one value for an attribute in an entity instance ensures the application of first normal form for the entity type.

Student_Course_Result Table

Student_Details			Course_Details				Results		
101	Davis	11/4/1986	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	82	A
102	Daniel	11/6/1987	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	62	C
101	Davis	11/4/1986	H6	American History		4	11/22/2004	79	B
103	Sandra	10/2/1988	C3	Bio Chemistry	Basic Chemistry	11	11/16/2004	65	B
104	Evelyn	2/22/1986	B3	Botany		8	11/26/2004	77	B
102	Daniel	11/6/1987	P3	Nuclear Physics	Basic Physics	13	11/12/2004	68	B
105	Susan	8/31/1985	P3	Nuclear Physics	Basic Physics	13	11/12/2004	89	A
103	Sandra	10/2/1988	B4	Zoology		5	11/27/2004	54	D
105	Susan	8/31/1985	H6	American History		4	11/22/2004	87	A
104	Evelyn	2/22/1986	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	65	B

Table in 1NF

Student_Course_Result Table

Student#	Student Name	Dateof Birth	Course#	CourseName	Pre Requisite	Duration InDays	DateOf Exam	Marks	Grade
101	Davis	04-Nov-1986	M4	Applied Mathematics	Basic Mathematics	7	11-Nov-2004	82	A
102	Daniel	06-Nov-1986	M4	Applied Mathematics	Basic Mathematics	7	11-Nov-2004	62	C
101	Davis	04-Nov-1986	H6	American History		4	22-Nov-2004	79	B
103	Sandra	02-Oct-1988	C3	Bio Chemistry	Basic Chemistry	11	16-Nov-2004	65	B
104	Evelyn	22-Feb-1986	B3	Botany		8	26-Nov-2004	77	B
102	Daniel	06-Nov-1986	P3	Nuclear Physics	Basic Physics	13	12-Nov-2004	68	B
105	Susan	31-Aug-1985	P3	Nuclear Physics	Basic Physics	13	12-Nov-2004	89	A
103	Sandra	02-Oct-1988	B4	Zoology		5	27-Nov-2004	54	D
105	Susan	31-Aug-1985	H6	American History		4	22-Nov-2004	87	A
104	Evelyn	22-Feb-1986	M4	Applied Mathematics	Basic Mathematics	7	11-Nov-2004	65	B

Second Normal Form

The second normal form rule is that the key attributes determine all non-key attributes.

A violation of second normal form occurs when there is a composite key, and part of the key determines some non-key attributes.

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

Second Normal Form

- STUDENT# is key attribute for Student,
- COURSE# is key attribute for Course
- STUDENT# COURSE# together form the composite key attributes for Results relationship.
- Other attributes like StudentName (Student Name), DateofBirth, CourseName, PreRequisite, DurationInDays, DateofExam, Marks and Grade are non-key attributes.

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

Student #, Course# -> Marks, Grade

Student# -> StudentName, DOB,

Course# -> CourseName, Prerequiste, DurationInDays

Course# -> Date of Exam

Second Normal Form - Tables in 2 NF

STUDENT TABLE

Student#	StudentName	DateofBirth
101	Davis	04-Nov-1986
102	Daniel	06-Nov-1987
103	Sandra	02-Oct-1988
104	Evelyn	22-Feb-1986
105	Susan	31-Aug-1985
106	Mike	04-Feb-1987
107	Juliet	09-Nov-1986
108	Tom	07-Oct-1986
109	Catherine	06-Jun-1984

COURSE TABLE

Course#	Course Name	Pre Requisite	Duration InDays
M1	Basic Mathematics		11
M4	Applied Mathematics	M1	7
H6	American History		4
C1	Basic Chemistry		5
C3	Bio Chemistry	C1	11
B3	Botany		8
P1	Basic Physics		8
P3	Nuclear Physics	P1	13
B4	Zoology		5

Second Normal form – Tables in 2 NF

Student#	Course#	Marks	Grade
101	M4	82	A
102	M4	62	C
101	H6	79	B
103	C3	65	B
104	B3	77	B
102	P3	68	B
105	P3	89	A
103	B4	54	D
105	H6	87	A
104	M4	65	B

Second Normal form – Tables in 2 NF

Exam_Date Table

Course#	DateOfExam
M4	11-Nov-04
H6	22-Nov-04
C3	16-Nov-04
B3	26-Nov-04
P3	12-Nov-04
B4	27-Nov-04

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.***

- STUDENT# and COURSE# are the key attributes.
- All other attributes, except grade are non-partially, non-transitively
- dependent on key attributes.

- **Student#, Course# - >** Marks
- **Marks ->** Grade

The third normal form rule is that the non-key attributes should be independent. This normal form is violated when there exists a dependency among non-key attributes in the form of a transitive dependency

3NF Tables

Student#	Course#	Marks
101	M4	82
102	M4	62
101	H6	79
103	C3	65
104	B3	77
102	P3	68
105	P3	89
103	B4	54
105	H6	87
104	M4	65

Third Normal Form – Tables in 3rd NF

MARKSGRADE TABLE

UpperBound	LowerBound	Grade
100	95	A+
94	85	A
84	70	B
69	65	B-
64	55	C
54	45	D
44	0	E

BCNF

Boyce-Codd Normal Form (BCNF)

The Boyce-Codd normal form (BCNF) extends the third normal form. **The Boyce-Codd normal form rule is that every determinant is a candidate key**. Even though Boyce-Codd normal form and third normal form generally produce the same result, *Boyce-Codd normal form is a stronger definition than third normal form.* Every table in Boyce-Codd normal form is by definition in third normal form.

Boyce-Codd normal form considers two special cases not covered by third normal form:

- Part of a composite entity identifier determines part of its attribute, and
- a non entity identifier attribute determines part of an entity identifier attribute.

These situations are only possible if there is a composite entity identifier, and dependencies exist from a non-entity identifier attribute to part of the entity identifier

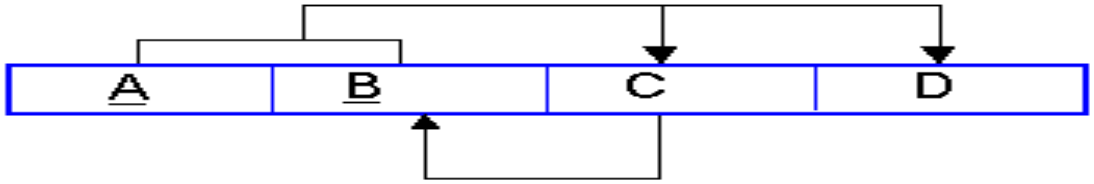
BCNF

Boyce Codd Normal Form (BCNF) is considered a special condition of third Normal form.

A table is in BCNF if every determinant is a candidate key.

A table can be in 3NF but not in BCNF. As discussed in previous slide this occurs **when non key attribute is a determinant of a key attribute.**

The dependency diagram may look like the one below:



The table is in 3NF. A and B are the keys and C and D depend on both A and B. There are no transitive dependencies existing between the non key attributes, C and D.

The table is not in BCNF because a dependency exists between C and B. In other words if we know the value of C we can determine the value of B.

BCNF

We can also show the dependencies as

$A\ B \rightarrow C\ D$

$C \rightarrow B$

Example:

An example table from the University Database might be as follows:

If we know the Student Number and Teacher Code we know the Offering(class) the student is in. We also know the review date for that student and teacher (Student progress is reviewed for that class by the teacher and student).

S_Num	T_Code	Offering#	Review Date
123599	FIT104	01764	2nd March
123599	PIT305	01765	12th April
123599	PIT107	01789	2nd May
346700	FIT104	01764	3rd March
346700	PIT305	01765 110	110110110110

BCNF

The dependencies are

S_Num, T_Code - > Offering#, Review Date
which means that the table is in third normal form.

The table is not in BCNF as if we know the offering number we know who the teacher is. Each offering can only have one teacher!
Offering# - > T_Code
A non key attribute is a determinant.

If we look at the table we can see a combination of T_Code and Offering# is repeated several times. Ex . FIT104 and 01764.

Converting to BCNF

The situation is resolved by following the steps below

1

The determinant, Offering#, becomes part of the key and the dependant attribute T_Code, becomes a non key attribute.
So the Dependency diagram is now
S_Num, Offering# → T_Code, Review Date

2 There are problems with this structure as T_Code is now dependant on only part of the key. This violates the rules for 2NF, so the table needs to be divided with the partial dependency becoming a new table.
The dependencies would then be
S_Num, Offering# → T_Code, Review Date
Offering# → T_Code

Converting to BCNF

3. The original table is divided into two new tables. Each is in 3NF and in BCNF.

StudentReview

S_Num	Offering#	Review Date
123599	01764	2nd March
123599	01765	12th April
123599	01789	2nd May
346700	01764	3rd March
346700	01765	7th May

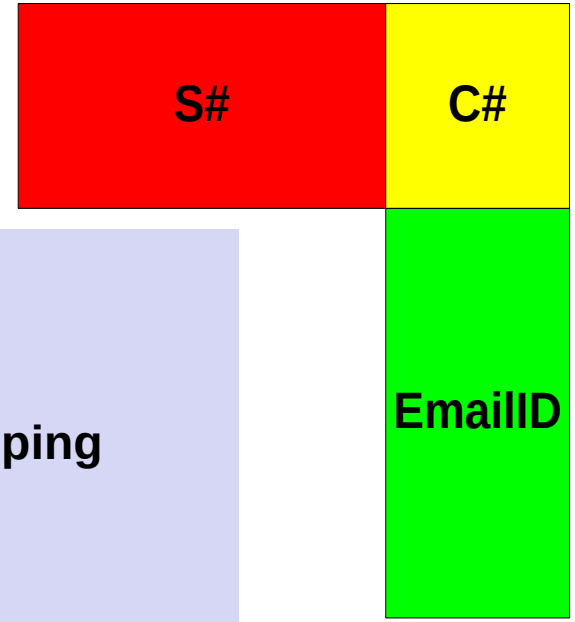
Offeringteacher

Offering#	T_Code
01764	FIT104
01765	PIT305
01789	PIT107

Consider this Result Table

Student#	EmailID	Course#	Marks
101	Davis@myuni.edu	M4	82
102	Daniel@myuni.edu	M4	62
101	Davis@myuni.edu	H6	79
103	Sandra@myuni.edu	C3	65
104	Evelyn@myuni.edu	B3	77
102	Daniel@myuni.edu	P3	68
105	Susan@myuni.edu	P3	89
103	Sandra@myuni.edu	B4	54
105	Susan@myuni.edu	H6	87
104	Evelyn@myuni.edu	M4	65

BCNF



Candidate Keys for the relation are

- **STUDENT# , COURSE#** and **COURSE# , EmailID**

Since **Course #** is overlapping, it is referred as **Overlapping Candidate Key.**

Functional Dependencies are

Student#, Course# - > Marks ,EmailID

EmailID - > Student# **(Non Key Determinant)**

Converting to BCNF

S# , Course# -> marks

EmailID -> S#

BCNF

STUDENT TABLE

EmailID	Student#
Davis@myuni.edu	101
Daniel@myuni.edu	102
Sandra@myuni.edu	103
Evelyn@myuni.edu	104
Susan@myuni.edu	105

BCNF Tables

Student#	Course#	Marks
101	M4	82
102	M4	62
101	H6	79
103	C3	65
104	B3	77
102	P3	68
105	P3	89
103	B4	54
105	H6	87
104	M4	65

Merits of Normalization

- Normalization is based on a mathematical foundation.
- Removes the redundancy to a greater extent. After 3NF, data redundancy is minimized to the extent of foreign keys.
- 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.

- ***Note: Instead of applying normalization principles during the relational design portion of logical database design phase, it is better to apply them during the conceptual modeling phase.***

Summary

- RDBMS handles data in the form of relations, tuples and fields
- Keys identify tuples uniquely
- ER modeling is a diagrammatic representation of the conceptual design of a database
- ER diagrams consist of Entity types, relationship types and attributes
- 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

Summary – contd.

- 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 – contd.

- 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 four 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 – contd.

- 3NF removes the transitive dependency
- BCNF removes dependency among key attributes
- Too much of normalization adversely affects SELECT or RETRIEVAL operations
- It is always better to normalize to 3NF for INSERT, UPDATE and DELETE intensive (On-line transaction) systems
- It is always better to restrict to 2NF for SELECT intensive (Reporting) systems
- While normalizing, use common sense and don’t use the normal forms as absolute measures

Thank You!

Prepared By: A. Srinivas Reddy

Email : digitech1993@gmail.com

Mobile : 9246115521