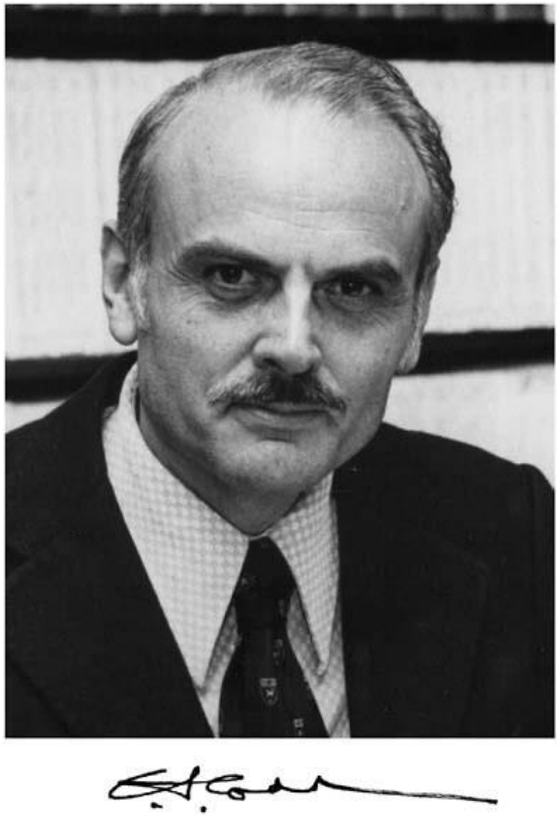
LESSON 3

DESIGN CONCEPTS

THE RELATIONAL DATABASE MODEL

Relational Data Model Brief History

- Was first introduced in 1970 by Edgar F. Codd
- Research projects were launched to prove the feasibility of the relational model
- IBM – developed System R
- University of California - Ingres

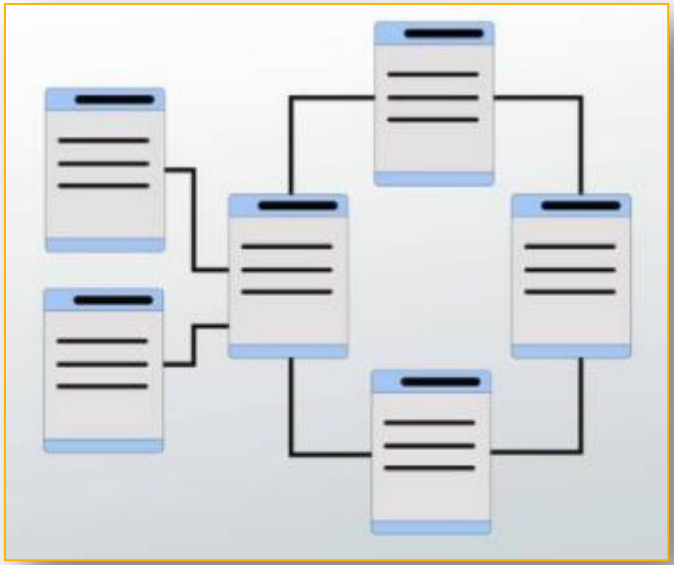


Relational Data Model Basic Definitions

- **Relations** refer to tables or data structures used to store and organize data. A relation is a fundamental concept in the relational model of databases.
- The relational model is the foundation for many modern DBMS, including popular ones like MySQL, PostgreSQL, Microsoft SQL Server, and Oracle Database.

Relational Data Model Basic Definitions

- Represents data in the form of tables
- Based on mathematical theory, therefore has a solid theoretical foundation
- Can be easily understood because of simple concepts involved.
- Consist of 3 components (Data Structure, Data Manipulation, Data Integrity)



Relational Data Model Components

- **Data Structure**
  - Data are organized in the form of tables, with rows and columns.

CUSTOMER

CUSTOMER_ID	LAST_NAME	FIRST_NAME	STREET	CITY	ZIP_CODE	COUNTRY
10302	Boucher	Leo	54, rue Royale	Nantes	44000	France
11244	Smith	Laurent	8489 Strong St	Las Vegas	83030	USA
11405	Han	James	636 St Kilda Road	Sydney	3004	Australia
11993	Mueller	Tomas	Berliner Weg 15	Tamm	71732	Germany
12111	Carter	Nataly	5 Tomahawk	Los Angeles	90006	USA
14121	Cortez	Nola	Av. Grande, 86	Madrid	28034	Spain
14400	Brown	Frank	165 S 7th St	Chester	33134	USA
14578	Wilson	Sarah	Seestreet #6101	Emory	1734	USA
14622	Jones	John	71 San Diego Ave	Arlington	69004	USA

Relational Data Model Components

- **Data Manipulation**
  - Powerful operations (using SQL Language) used to manipulate data stored in the relations.

OUTER QUERY — **SUBQUERY OR INNER QUERY**

```
SELECT EmployeeName, EmergencyContactName
FROM Employee_Info
WHERE Address IN (SELECT Address
                  FROM Office
                  WHERE Country  = "INDIA")
```

edureka!

Relational Data Model Components

- **Data Integrity**
  - Mechanisms to specify business rules that maintain the integrity of data when they are manipulated.

Constraint Type

- NOT NULL
- Unique key
- Primary key
- Foreign key

Relational Data Model Components

Table 5-1 Types of Integrity Constraints

Constraint Type	Description	See Also
`NOT NULL`	Allows or disallows inserts or updates of rows containing a **null** in a specified column.	"NOT NULL Integrity Constraints"
Unique key	Prohibits multiple rows from having the same value in the same column or combination of columns but allows some values to be null.	"Unique Constraints"
Primary key	Combines a `NOT NULL` constraint and a unique constraint. It prohibits multiple rows from having the same value in the same column or combination of columns and prohibits values from being null.	"Primary Key Constraints"
Foreign key	Designates a column as the foreign key and establishes a relationship between the foreign key and a primary or unique key, called the **referenced key**.	"Foreign Key Constraints"
Check	Requires a database value to obey a specified condition.	"Check Constraints"
`REF`	Dictates types of data manipulation allowed on values in a `REF` column and how these actions affect dependent values. In an object-relational database, a built-in data type called a `REF` encapsulates a reference to a row object of a specified object type. Referential integrity constraints on `REF` columns ensure that there is a row	*Oracle Database Object-Relational Developer's Guide* to learn about `REF` constraints

Relational Data Structure

- **Relation**
  - A named two-dimensional table of data.
  - Each relation (or table) consists of a set of named columns and an arbitrary number of unnamed rows.
  - Column -> Attribute, Rows -> Records
  - Shorthand notation (Attributes should be separated by comma (,)):
  - NameofRelation (Names of attributes)
  - Example:

```
EMPLOYEE1(EmpID, Name, DeptName, Salary)
```

Relation **Attributes**

Relational Keys

- **Primary Key**
  - an attribute or a combination of attributes that uniquely identifies each row in a relation.
  - Designated by underlining the attribute name(s)
  - See below example:

```
EMPLOYEE1(EmpID, Name, DeptName, Salary)
```

Primary Key

Relational Keys

- **Foreign Key**
  - is an attribute in a relation that serves as the primary key of another relation.
  - See below example:

```
EMPLOYEE1(EmpID, Name, DeptName, Salary)
DEPARTMENT(DeptName, Location, Fax)
```

Primary Key

```
EMPLOYEE1(EmpID, Name, DeptName, Salary)
```

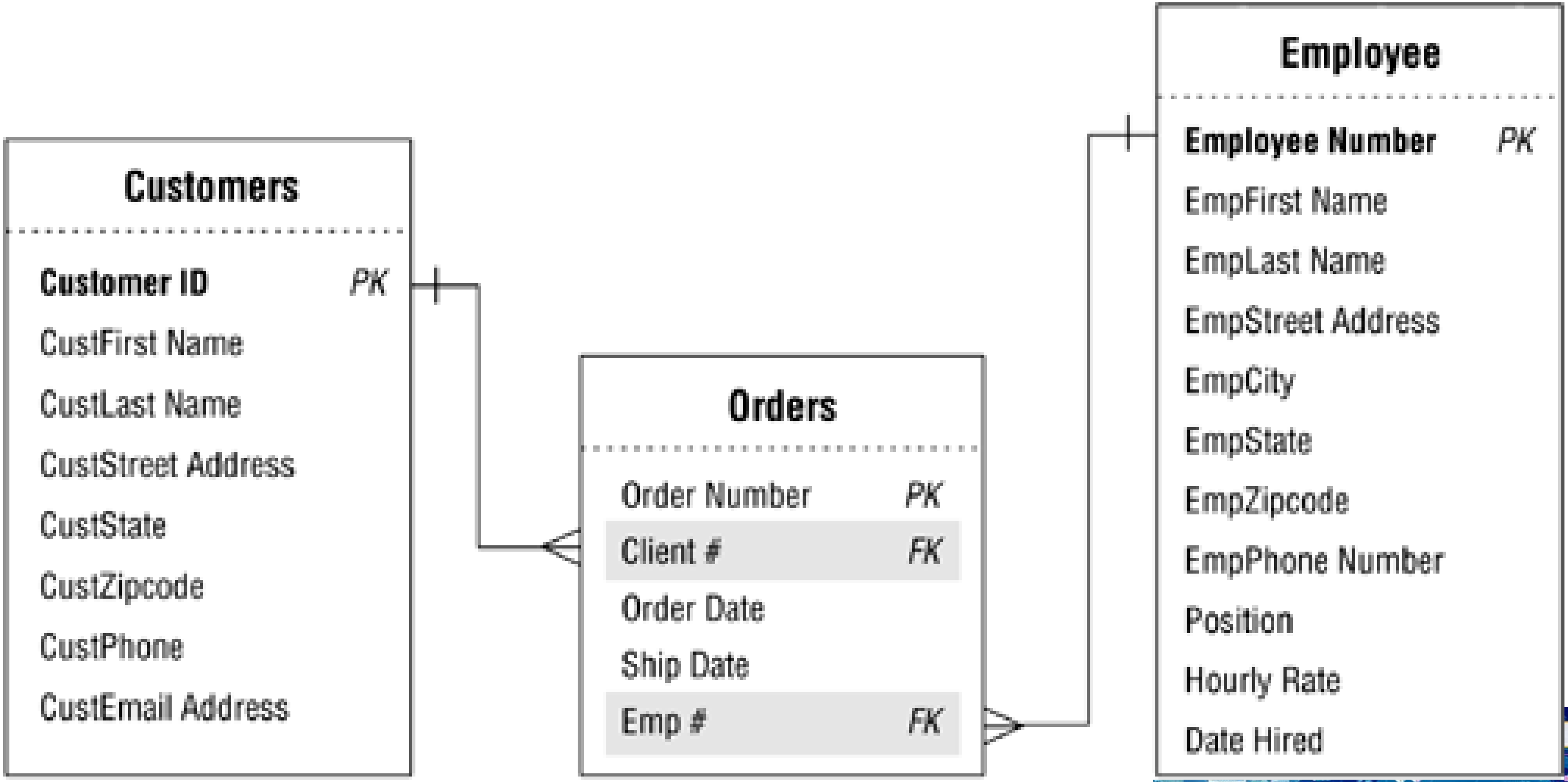
Foreign Key

```
Student(sID, sLname, sFname, sMI, pID, sBirthdate)
```

```
Program(pID, pName, pCollege, pDuration, pCuri)
```

```
Grades(gID, sID, subjName, sGrade)
```

Relational Keys



Student

sID	sLname	sFname	sMI	pID	sBirthdate
2008613	Frondez	Alfred	B	1001	10/12/1999
2008614	Vasquez	Lourdes	B	1002	07/25/1998
2008615	Apondar	Shaina Mae	B	1001	04/23/2000
2008615					

Program

pID	pName	pCollege	pDuration	pCuri
1001	BSIT	CCS	4 years	2015
1002	BSME	COE	5 years	2018

```
Student(sID, sLname, sFname, sMI, pID, sBirthdate)
```

```
Program(pID, pName, pCollege, pDuration, pCuri)
```

```
Grades(gID, sID, subjName, sGrade)
```

Properties of Relations

Not all tables are relations. Relations have several properties that distinguish them from non-relational tables. Such as:

- Each relation (or table) in a database has a **unique name**.
- An entry at the intersection of each row and column is atomic (or single valued). **There can be only one value associated** with each attribute on a specific row of a table; **no multivalued attributes are allowed** in a relation.
- Each row is unique; **no two rows in a relation can be identical**.

Properties of Relations

- **Each attribute** (or column) within a table has a **unique name**.
- The sequence of columns (left to right) is insignificant. The **order of the columns** in a relation **can be changed** without changing the meaning or use of the relation.
- **The sequence of rows (top to bottom) is insignificant**. As with columns, the order of the rows of a relation may be changed or stored in any sequence.

Removing Multivalued Attributes from Tables

- a table that contains one or more multivalued attributes is not a relation.

EmpID	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Yang Yang	Marketing	48,000	SPSS	6/19/201X
				Surveys	10/7/201X
140	Xian Li	Accounting	52,000	Tax Acc	12/8/201X
110	Lisa Manoban	Info Systems	43,000	Visual Basic	1/12/201X
				C++	4/22/201X
190	Yitian Hu	Finance	55,000		
150	Yi Lin	Marketing	42,000	SPSS	6/16/201X
				Java	8/12/201X

Sample Table with Multivalued attributes

Removing Multivalued Attributes from Tables

EmpID	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Yang Yang	Marketing	48000	SPSS	6/19/201X
100	Yang Yang	Marketing	48000	Surveys	10/7/201X
140	Xian Li	Accounting	52,000	Tax Acc	12/8/201X
110	Lisa Manoban	Info Systems	43,000	Visual Basic	1/12/201X
110	Lisa Manoban	Info Systems	43,000	C++	4/22/201X
190	Yitian Hu	Finance	55,000		
150	Yi Lin	Marketing	42,000	SPSS	6/16/201X
150	Yi Lin	Marketing	42,000	Java	8/12/201X

Sample Table fixed with Multivalued attributes

Removing Multivalued Attributes from Tables

1639	George	Barnes	reading
5629	Susan	Noble	hiking, movies
3388	Erwin	Star	hockey, skiing
5772	Alice	Buck	
1911	Frank	Borders	photography, travel, art
4848	Hanna	Diedrich	gourmet cooking

Removing Multivalued Attributes from Tables

1639	George	Barnes
5629	Susan	Noble
3388	Erwin	Star
5772	Alice	Buck
1911	Frank	Borders
4848	Hanna	Diedrich

1639	reading
5629	hiking
5629	movies
3388	hockey
3388	skiing
1911	photography
1911	travel
1911	art
4848	gourmet cooking

Relational Database

- may consist of any number of relations.
- The structure of the database is described through the use of a schema
- **Schema**
  - a description of the overall logical structure of the database

Relational Database Schema

- two common methods for expressing a schema:
- Short text statements, in which each relation is named and the names of its attributes follow in parentheses.

```
EMPLOYEE(EmpID, Name, DeptName, Salary)
DEPARTMENT(DeptName, Location, Fax)
```

- A graphical representation, in which each relation is represented by a rectangle containing the attributes for the relation.

EMPLOYEE

EmpID	Name	DeptName	Salary

DEPARTMENT

DeptName	Location	Fax

Integrity Constraints

- rules limiting acceptable values and actions, whose purpose is to facilitate maintaining the accuracy and integrity of data in the database.
- Major types of integrity constraints:
  - Domain Constraints
  - Entity Integrity
  - Reference Integrity

Domain Constraints

- All of the values that appear in a column of a relation must be from the same domain.
- A domain is the set of values that may be assigned to an attribute
- A domain definition usually consists of the following components: domain name, meaning, data type, size (or length), and allowable values or allowable range (if applicable).

IDNum	Name	Course	BirthDate	Age	ContactNum
1001	Lopez, Ana	BSIT	10/21/1992	27	34510245678
1002	Cruz, Tom	BLIS	08/03/1991	29	02394517523
1003	Morgan, Kate	BSIS	05/23/1994	26	NONE
1004	Gomez, Tom	BSIT	07/18/1993	Y	19375428917
1005	Hoffer, Maine	BSCS	May 12, 1993	27	N/A

*****Encircled data: Not allowed for they do not follow the datatype allowed for the column.**

Entity Integrity

- designed to ensure that every relation has a primary key and that the data values for that primary key are all valid.
- it guarantees that **every primary key attribute is non-null**.
- **Null** - A value that may be assigned to an attribute when no other value applies or when the applicable value is unknown.
- **Entity integrity rule**
  - A rule that states that no primary key attribute (or component of a primary key attribute) may be null.

Entity Integrity

- A rule that states that either each foreign key value must match a primary key value in another relation or the foreign key value must be null.
- maintains consistency among the rows of two relations

EMPLOYEE

EmpID	Name	DeptName	Salary

DEPARTMENT

DeptName	Location	Fax

- In above example, the value of the “DeptName” field in the EMPLOYEE table must be referenced from the “DeptName” in the DEPARTMENT table. If it does not apply, value must ben null.

THE E - R MODEL


CCIT 105 - INFORMATION MANAGEMENT 1
LESSON 3
DESIGN
CONCEPTS

The E-R Model

- Entity Relationship Model also called e-r model and it is a high level model
- This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

What is an Entity Relationship Diagram (ERD)?

- ERD is a data modeling technique used in software engineering to produce a conceptual data model of an information system.
- So, ERDs illustrate the logical structure of databases.
- The major activity of this phase is identifying entities, attributes, and their relationships to construct model using the Entity Relationship Diagram.
- Entity -> table
- Attribute -> column
- Relationship -> line

Starting an ERD

- Define the Entities.
- Define the Relationships.
- Add attributes to the relationships.
- Add cardinality to the relationships.
- Don’t forget to use proper naming conventions and symbol representation.
- Layout the diagram with minimal line crossing.
- Place subject entity types on the top of the diagram.

Starting an ERD

- Place plural entity types below a single entity type in a one-to-many relationship.
- Place entity types participating in one-to-one and many-to-many relationships alongside each other.
- Group closely related entity types when possible. Try to keep the length of relationship lines as short as possible. Also try to minimize the number of changes of direction in a single line.
- Show the most relevant relationship name. One name must always be shown.

Building Blocks of ERD

Type	English Grammar Equivalent	Example
Entity	Proper Noun	Student, Employee, Instructor, Courses, Room
Relationship	Verb	has, teaches, belongs, handles
Attribute	Adjective	Height, Age, Gender, Nationality, First name

E-R MODEL NOTATION

Chen Notation

- A popular standard and is widely used worldwide in database and software design.
- Dr. Peter Chen's entity-relationship model is based on a ***natural view*** of how the real world comprises entities and their relationships.
- It was created to combine the three entity-data models (network model, relationship model, entity model).

Components of Chen Notation

Entity - Represented by rectangles

In a database, an entity is normally represented by a table.

Attribute - Represented by ovals containing attribute's name

An attribute is a property or characteristic of an entity.

Relationship - Represented by a diamond shape

These shapes are used together to show the relationship between tables - how they interact.

Chen Notation - ENTITY

- An entity may be any object, class, person or place and has a noun name.
- be represented as rectangles in ERD.

- Some examples of each of these kinds of entities follow:
  - Person: EMPLOYEE, STUDENT, PATIENT
  - Place: STORE, WAREHOUSE, STATE
  - Object: MACHINE, BUILDING, AUTOMOBILE
  - Event: SALE, REGISTRATION, RENEWAL
  - Concept: ACCOUNT, COURSE, WORK CENTER

Chen Notation - ATTRIBUTE

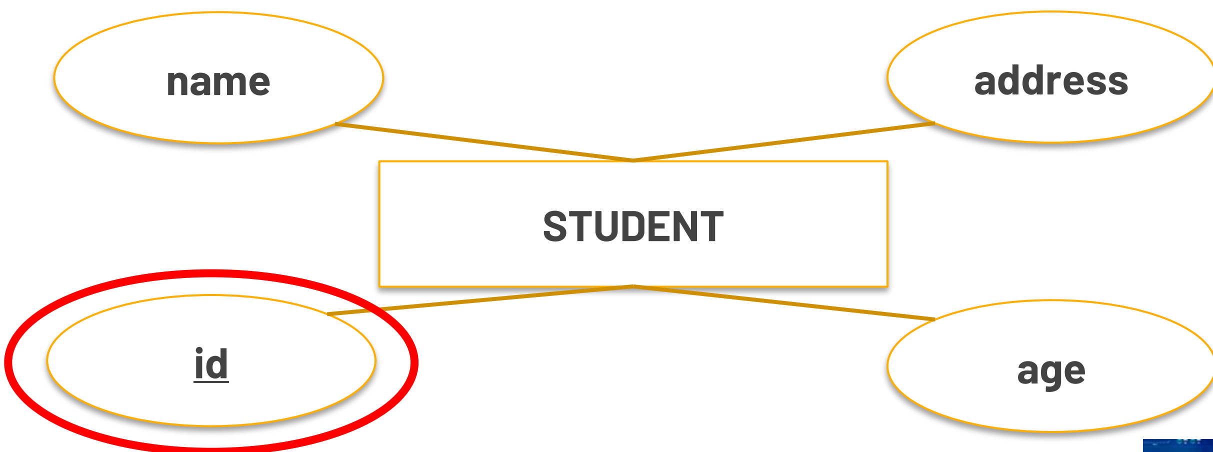
- a property or characteristic of an entity type that is of interest to the organization (often corresponds to a field in a table).
- Example:

STUDENT	Student ID, Student Name, Home Address, Phone Number, Major
AUTOMOBILE	Vehicle ID, Color, Weight, Horsepower
EMPLOYEE	Employee ID, Employee Name, Payroll Address, Skill

- In naming attributes, we use an initial capital letter followed by lowercase letters.
- If an attribute name consists of more than one words, we use a space between the words and we start each word with a capital letter.

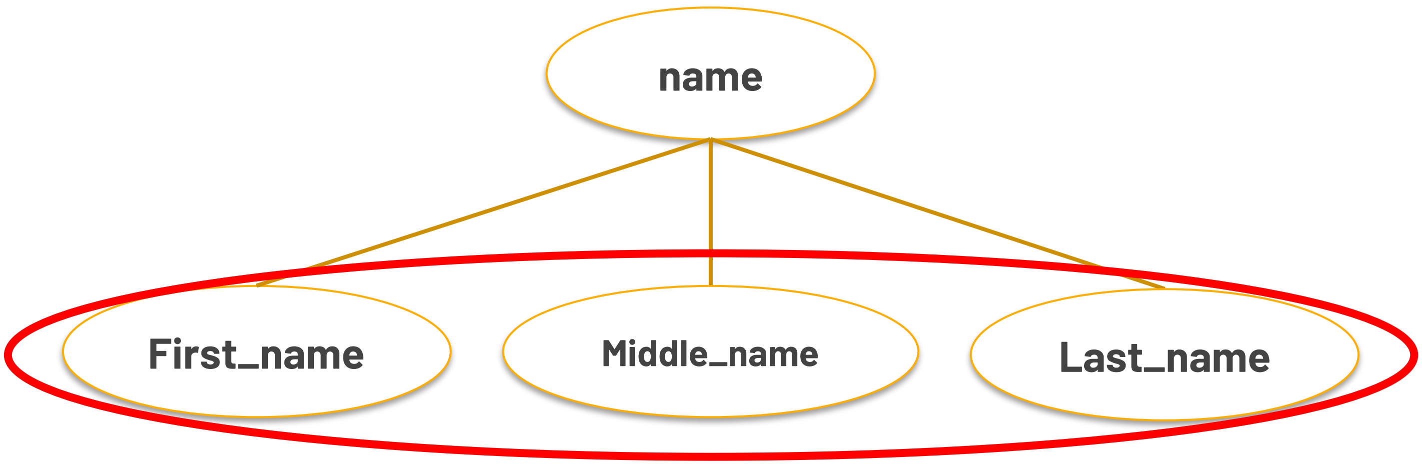
Chen Notation - ATTRIBUTE

- **Key Attribute**
  - used to represent the main characteristics of an entity (represents a primary key).
  - Represented by an oval shape with the text underlined.



Chen Notation - ATTRIBUTE

- **Composite Attribute**
  - An attribute that composed of many other attributes is known as a composite attribute.

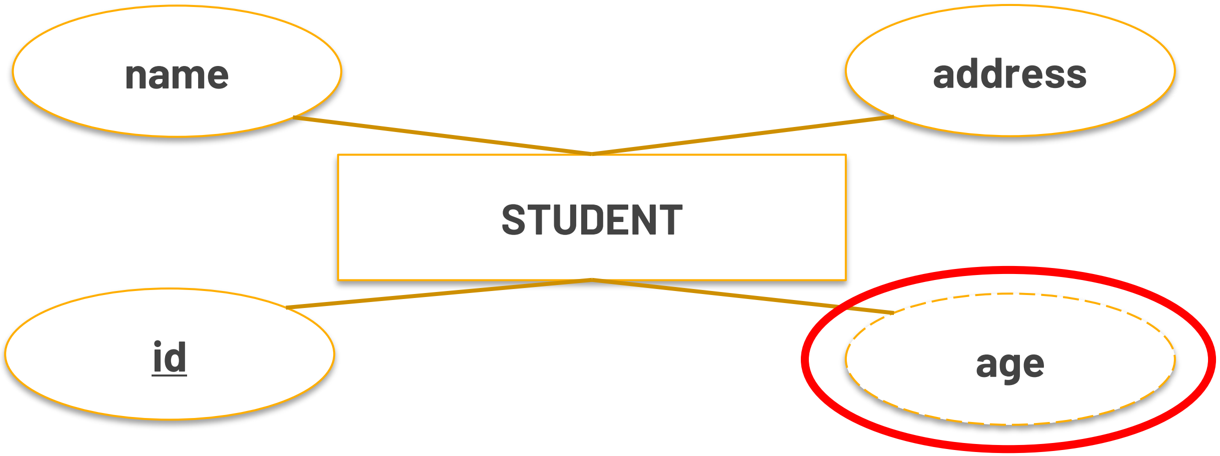


Chen Notation - ATTRIBUTE

- **Multivalued Attribute**
  - An attribute can have more than one value.
  - The double oval is used to represent multivalued attribute.

Chen Notation - ATTRIBUTE

- **Derived Attribute**
  - An attribute that can be derived from other attribute is known.
  - It can be represented by a dashed oval.



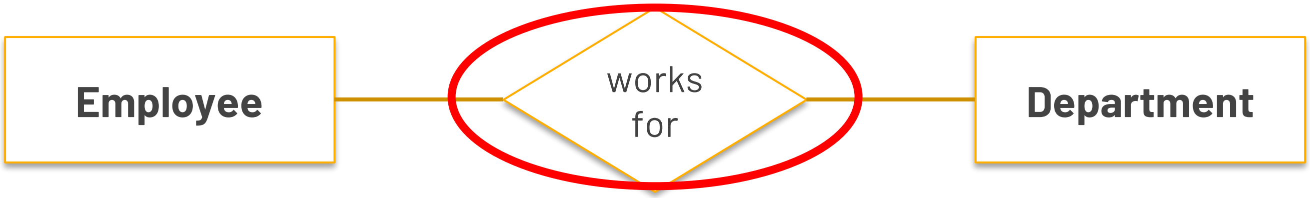
Chen Notation - ATTRIBUTE

- **Required attribute**
  - An attribute that must have a value for every entity (or relationship) instance with which it is associated.

- **Optional attribute**
  - An attribute that may not have a value for every entity (or relationship) instance with which it is associated.

Chen Notation - RELATIONSHIP

- A relationship is used to describe the relation between entities.
- represent the most complex business rules shown in an ERD.
- Diamond shape is used to represent to relationship.



Chen Notation - RELATIONSHIP

- **Cardinality**
- The degree of relationship (cardinality) is represented by characters "**1**", "**N**" or "**M**" usually placed at the ends of the relationships:

Types of Relationship

1. one-to-one (1:1)
2. one-to-many (1:N)
3. many-to-one (N:1)
4. many-to-many (M:N)

Chen Notation - RELATIONSHIP

- **one-to-one (1:1)**

The employee can manage only one department, and each department can be managed by one employee only:

Chen Notation - RELATIONSHIP

- **one-to-many (1:N)**

The customer may place many orders, but each order can be placed by one customer only:

Chen Notation - RELATIONSHIP

- **many-to-one (N:1)**

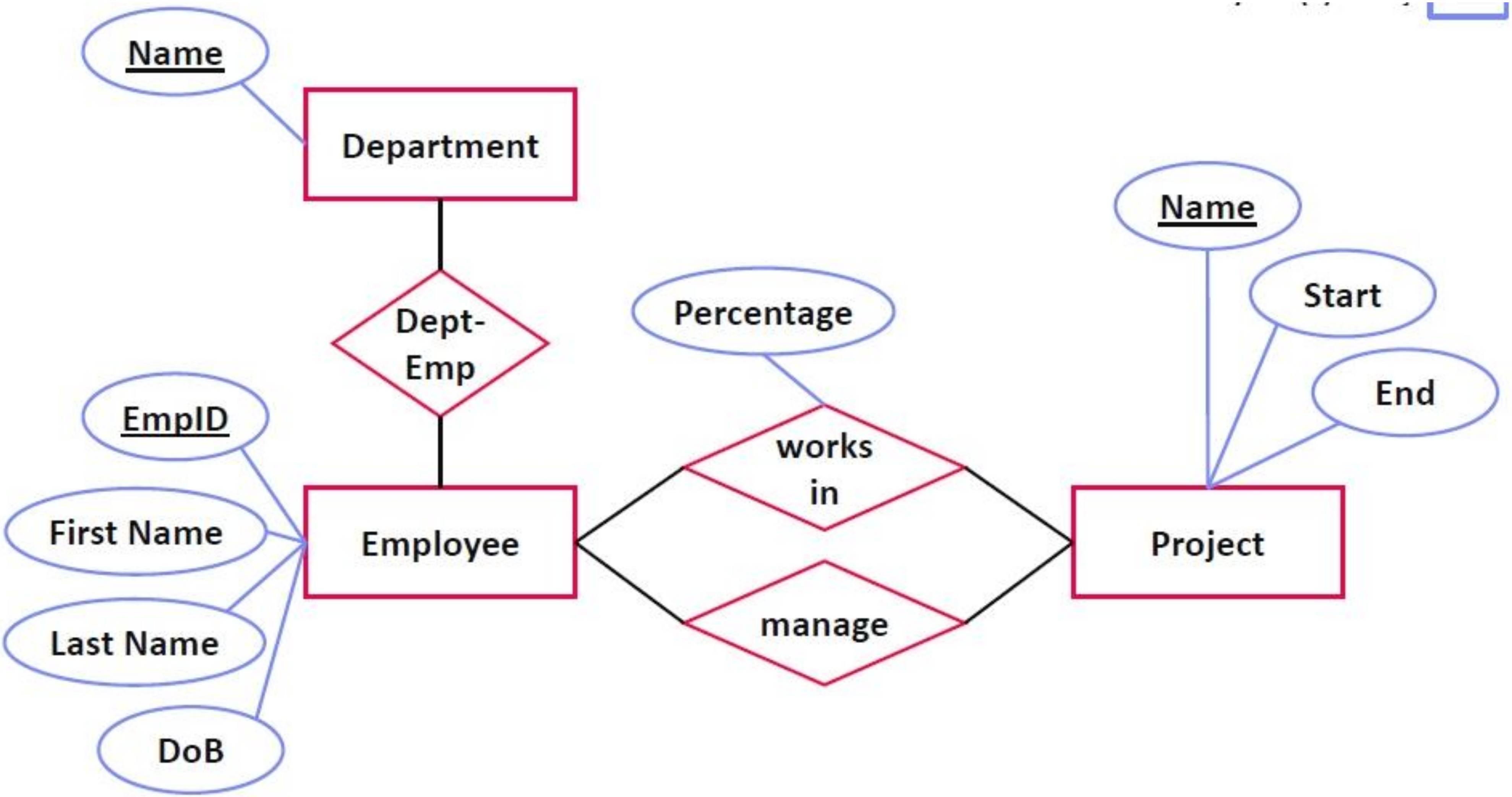
Many employees may belong to one department, but one particular employee can belong to one department only:

Chen Notation - RELATIONSHIP

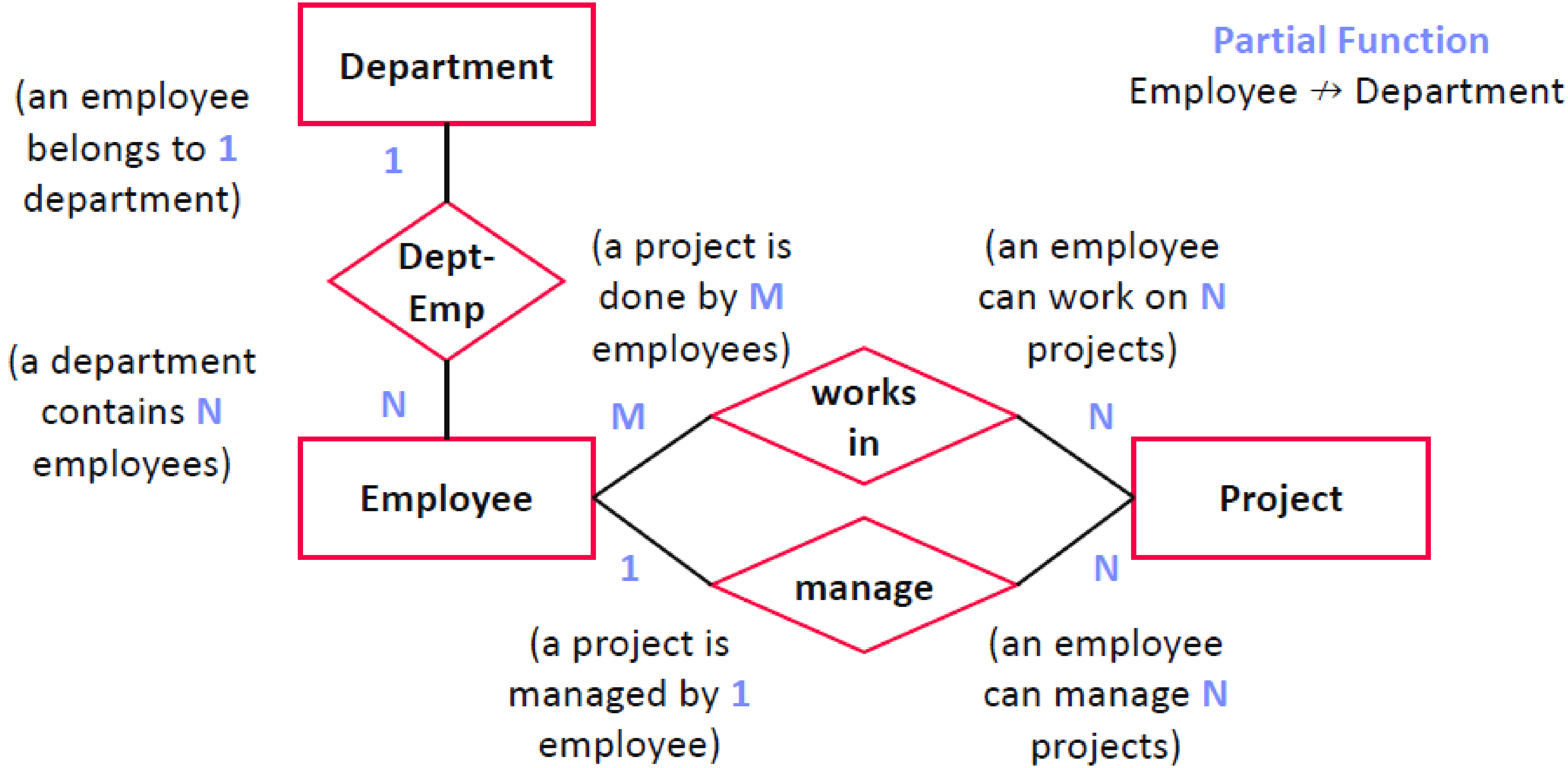
- **many-to-many (M:N)**

One student may belong to more than one student organizations, and one organization can admit more than one student:

Chen Notation - Example



Chen Notation - Example



Crow’s Foot Notation

- Known as IE notation (most popular)
- A type of cardinality notation. It is called crow's foot notation because of the shapes, which include circles, bars, and symbols, that indicate various possibilities.
- A single bar indicates one, a double bar indicates one and only one, a circle indicates zero and a crow's foot indicates many.

DESIGN CONCEPTS

Crow's Foot Notation - ENTITY

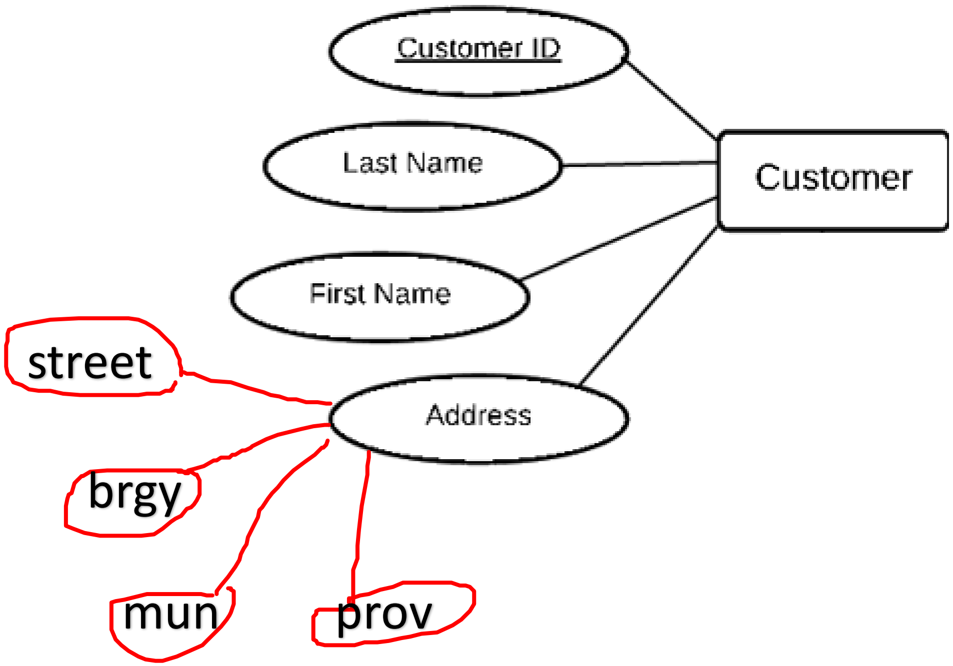
- Represented by a rectangle, with its name on the top. The name is singular (entity) rather than plural (entities).

Crow's Foot Notation - ATTRIBUTES

- An attribute is a property that describe a particular entity.
- The attribute(s) that uniquely distinguishes an instance of the entity is the ***identifier***. Usually, this type of attribute is marked with an asterisk.

Entity
* s_id s_fname s_lname s_dateofbirth s_program

Attributes



Customer

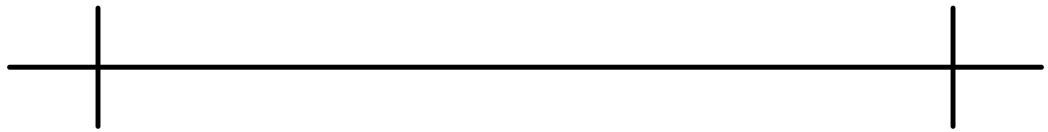
* Customer_ID (PK)
Last_Name
First_Name
Street
Brgy
Mun
prov

Chen Notation

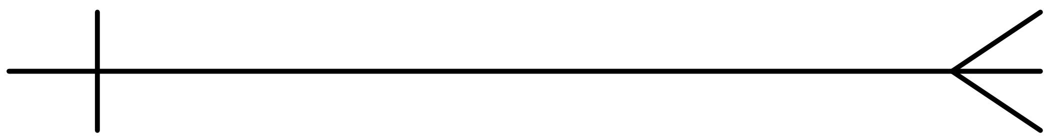
Crows Foot Notation

Crow's Foot Notation - RELATIONSHIP

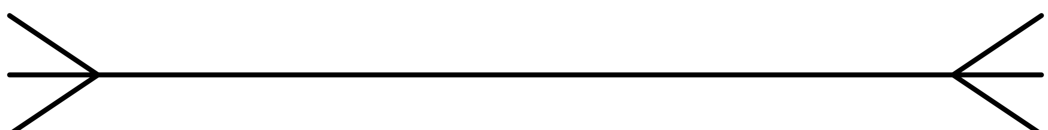
- **Types of Cardinality**
  - 1-to-1 relationship



  - 1-to-M relationship



  - M-to-N relationship



Crow's Foot Notation - RELATIONSHIP

- **Cardinality Constraints**
  - Mandatory One

  - Mandatory Many

  - Optional One

  - Optional Many

ACTIVITY

- A musician might have created none or arbitrary many albums, and any album is created by at least one musician.
- Every musician has exactly one agent, and an agent might be responsible for one to ten musicians.
- Every musician occupies exactly one studio, and musicians never share a studio.

- Studio – (Country, City, Size)
- Musician – (<u>MID</u>, Name, URL)
- Albums – (<u>AID</u>, Year, Name)
- Agent – (<u>AgID</u>, Name)

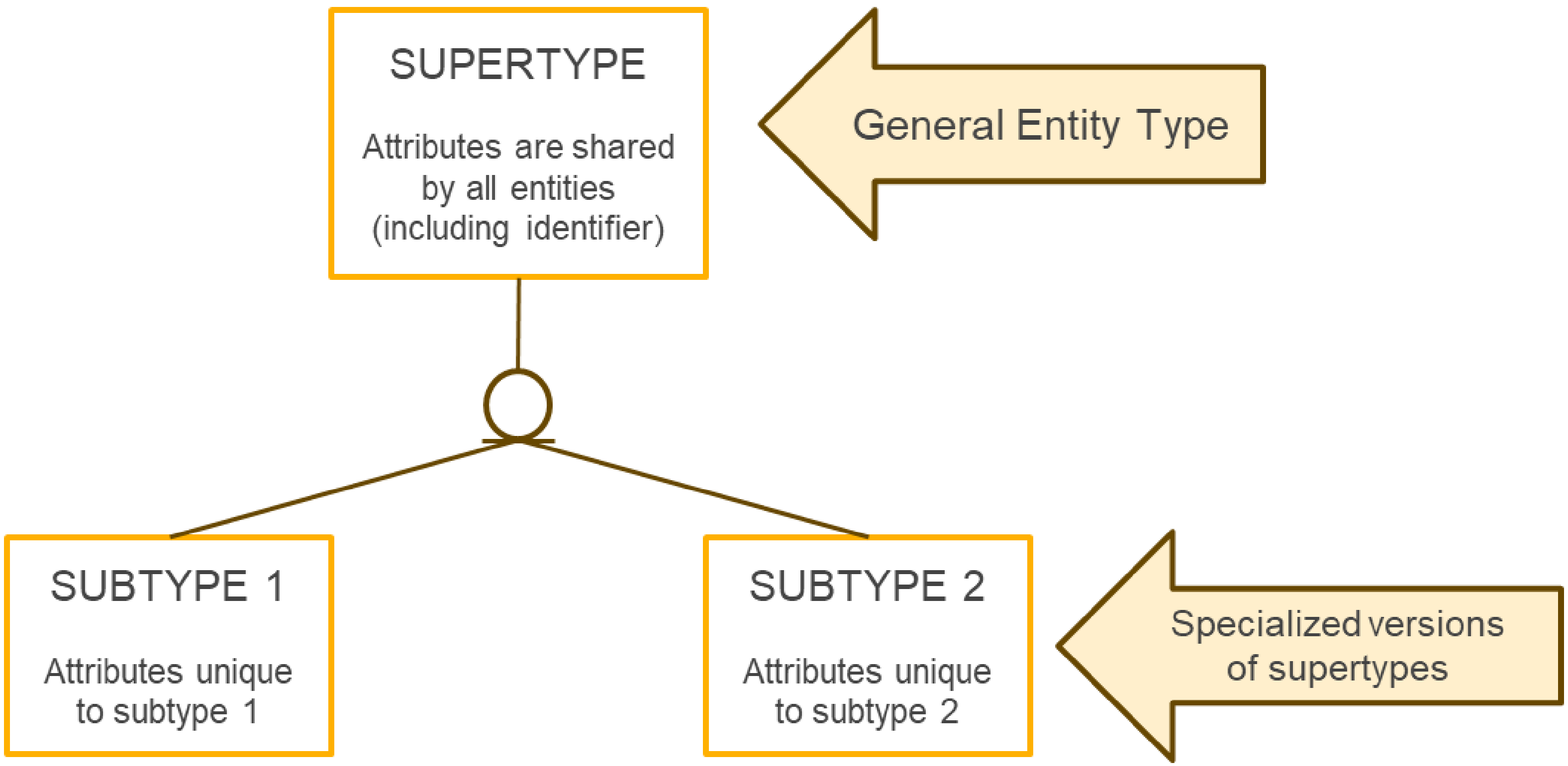
\- occupied, created, has

ADVANCED DATA MODELLING

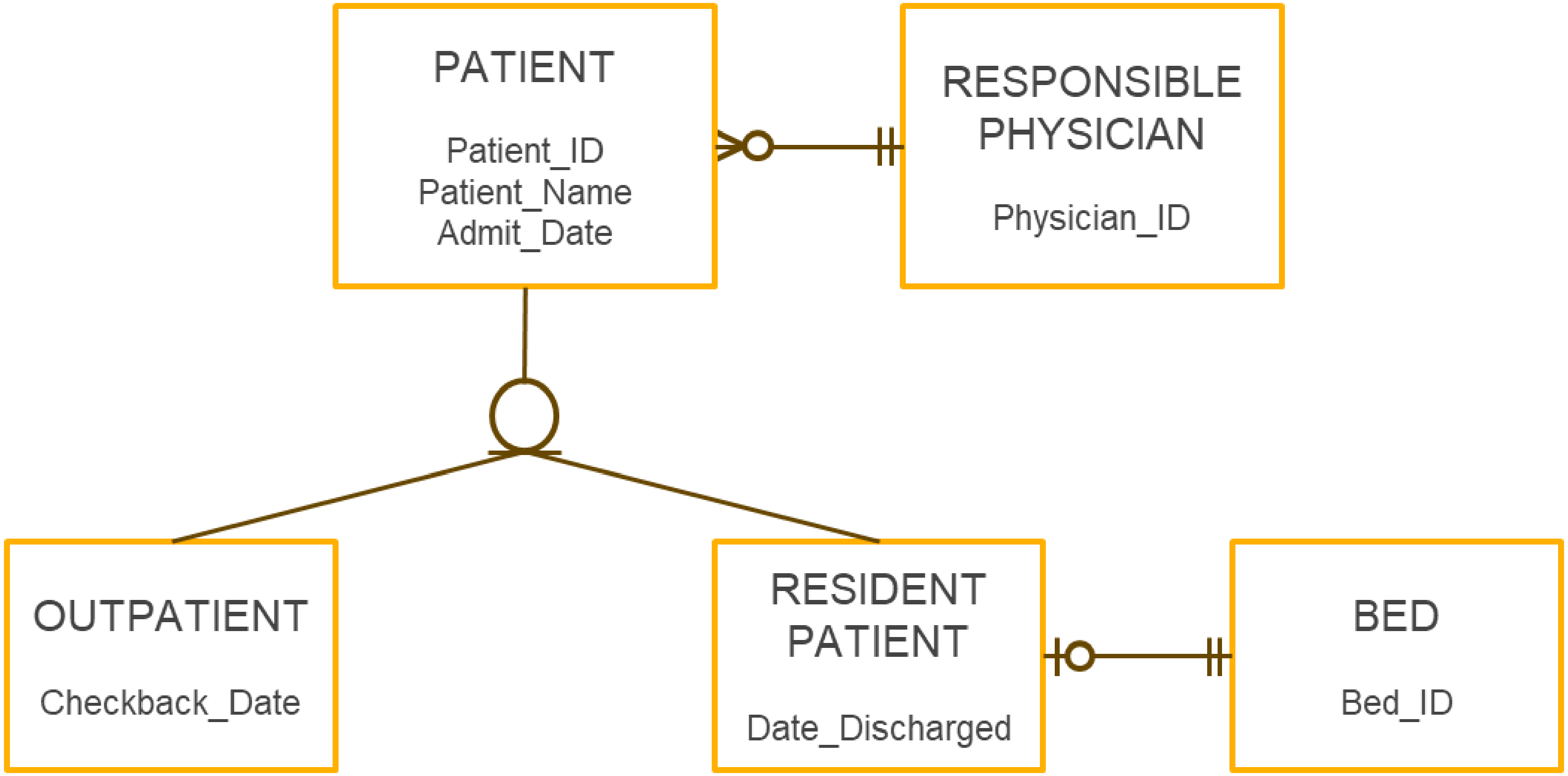
Supertype & Subtype

- An entity can be divided into several subgroups or subtypes
- An entity that can be divided into subtypes is called **super type**
- **Subtype**: A subgrouping of the entities in an entity type that has attributes distinct from those in other subgroupings
- **Supertype**: A generic entity type that has a relationship with one or more subtypes
- **Attribute Inheritance:**
  - Subtype entities inherit values of all attributes of the supertype
  - An instance of a subtype is also an instance of the supertype

Basic Notation for Supertype/Subtype



Supertype/subtype Relationships in a Hospital

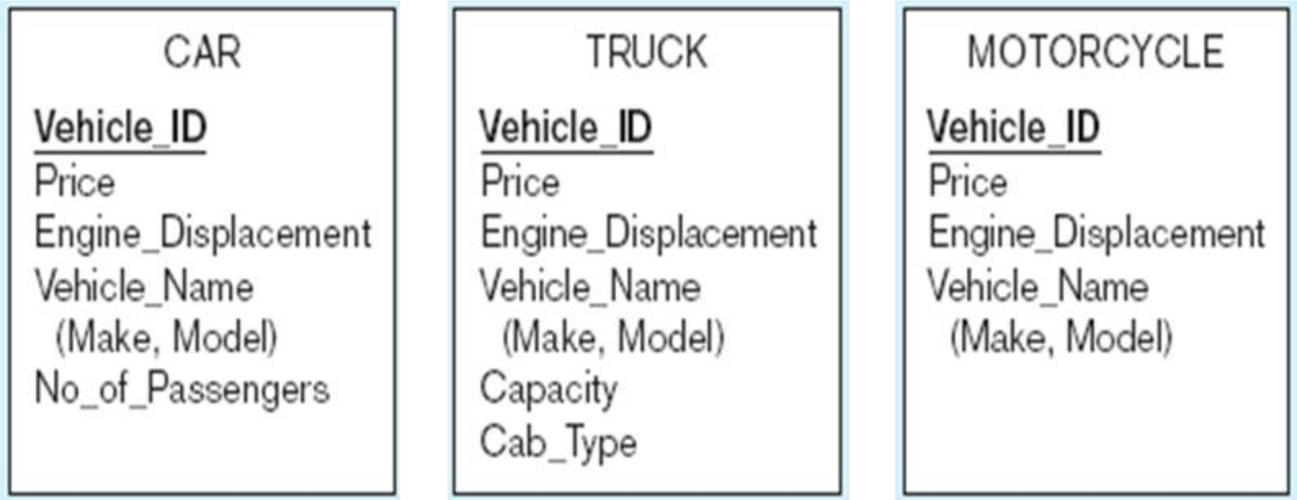


Generalization and Specialization

- **Generalization**: The process of defining a more general entity type from a set of more specialized entity types. **BOTTOM-UP**
- **Specialization**: The process of defining one or more subtypes of the supertype and forming supertype/subtype relationships. **TOP-DOWN**

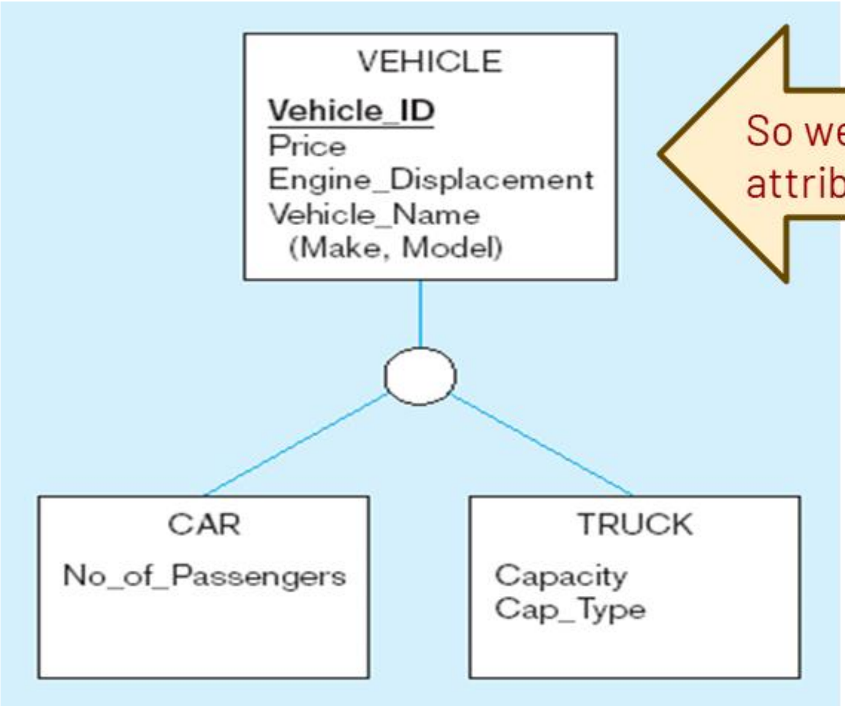
Generalization

- Three entity types: CAR, TRUCK, and MOTORCYCLE



Generalization

- Generalization to VEHICLE supertype

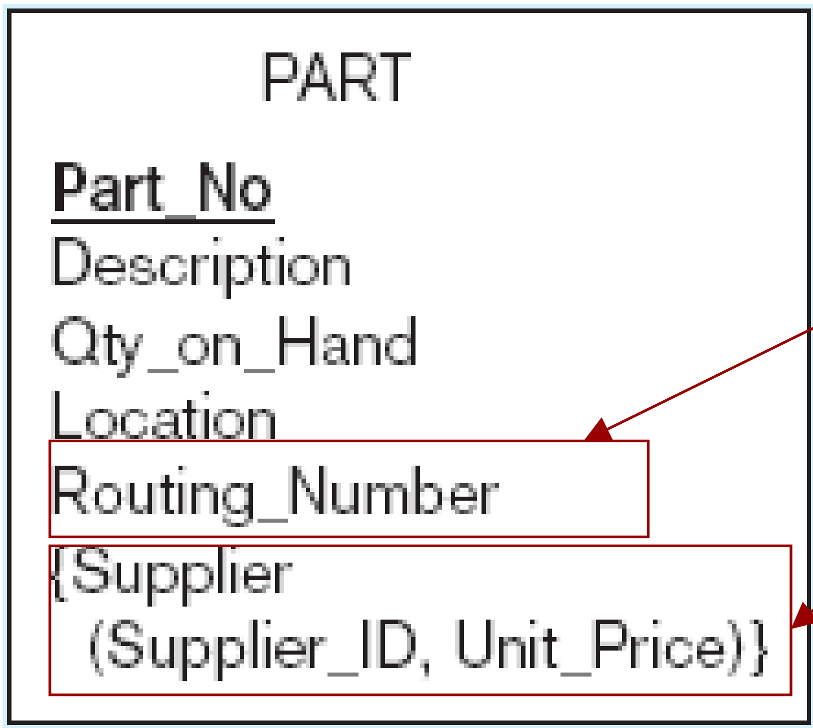


So we put the shared attributes in a supertype

Note: no subtype for motorcycle, since it has no unique attributes

Specialization

- Entity type PART

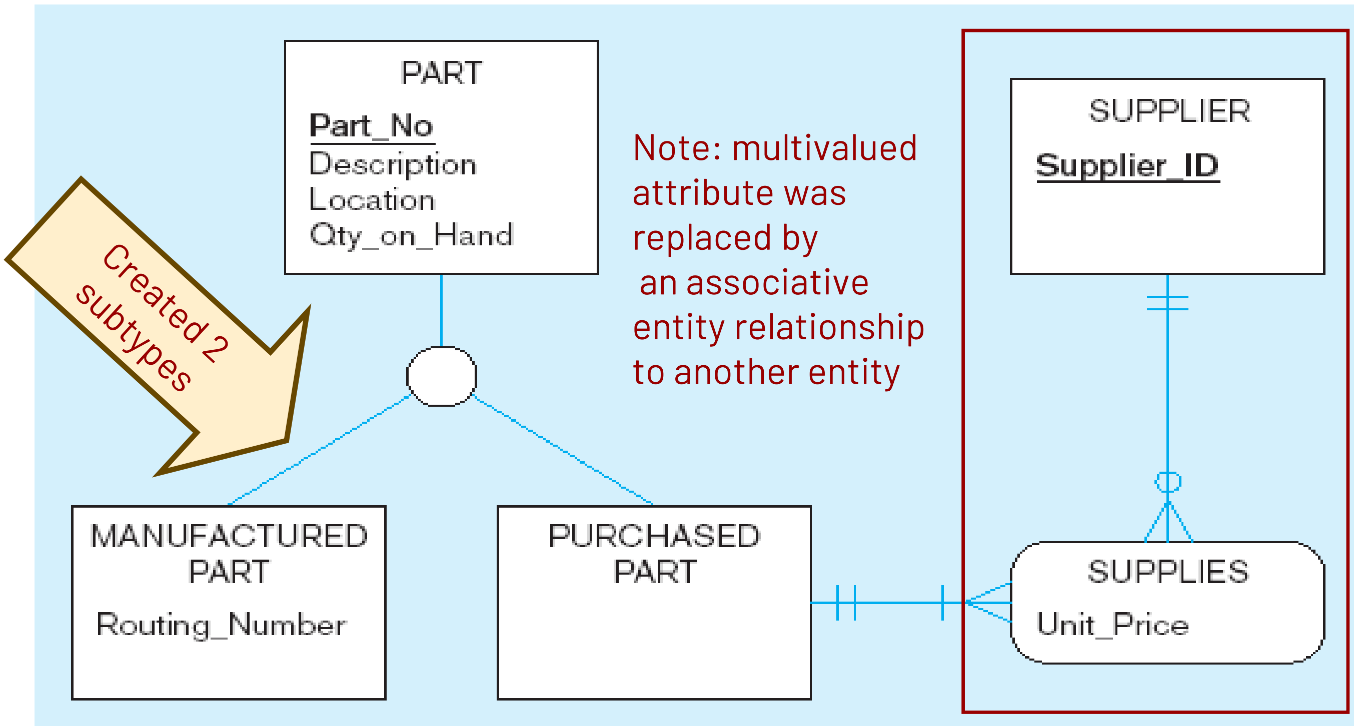


Only applies to manufactured parts

Applies only to purchased parts

Specialization

- Specialization to MANUFACTURED PART and PURCHASED PART

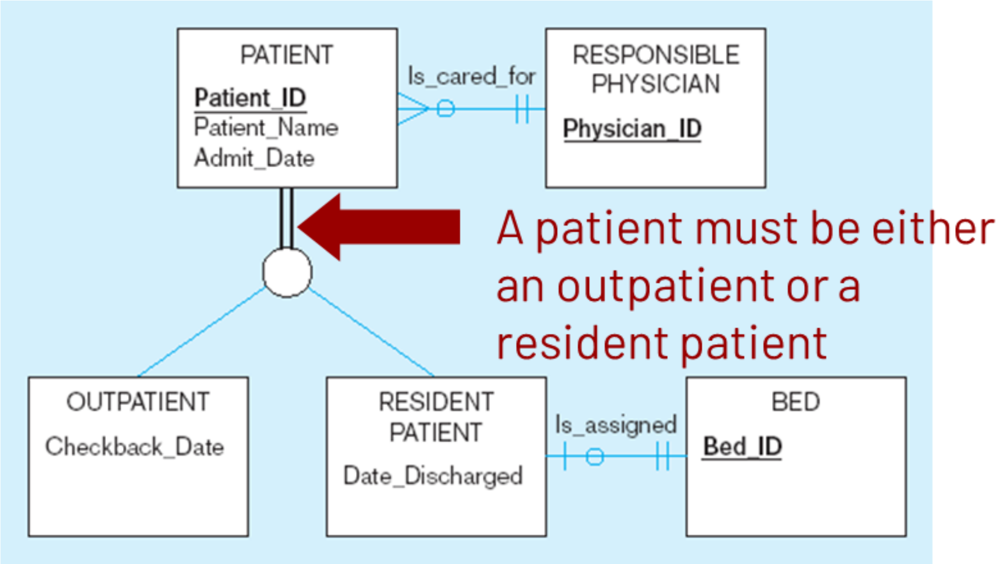


Constraints in Supertype/ Completeness Constraint

- **Completeness Constraints**: Whether an instance of a supertype ***must*** also be a member of at least one subtype
- Total Specialization Rule: Yes (double line)
- Partial Specialization Rule: No (single line)

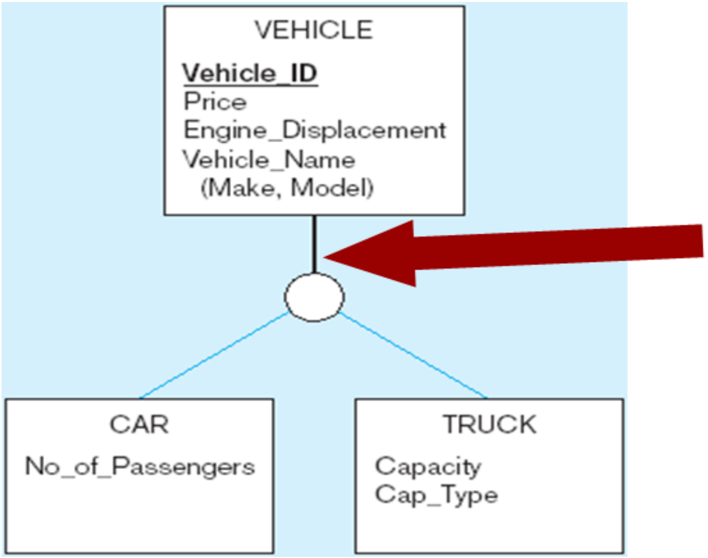
Constraints in Supertype/ Completeness Constraint

- Examples of completeness constraints
  - Total specialization rule



Constraints in Supertype/ Completeness Constraint

- Examples of completeness constraints
  - Partial specialization rule



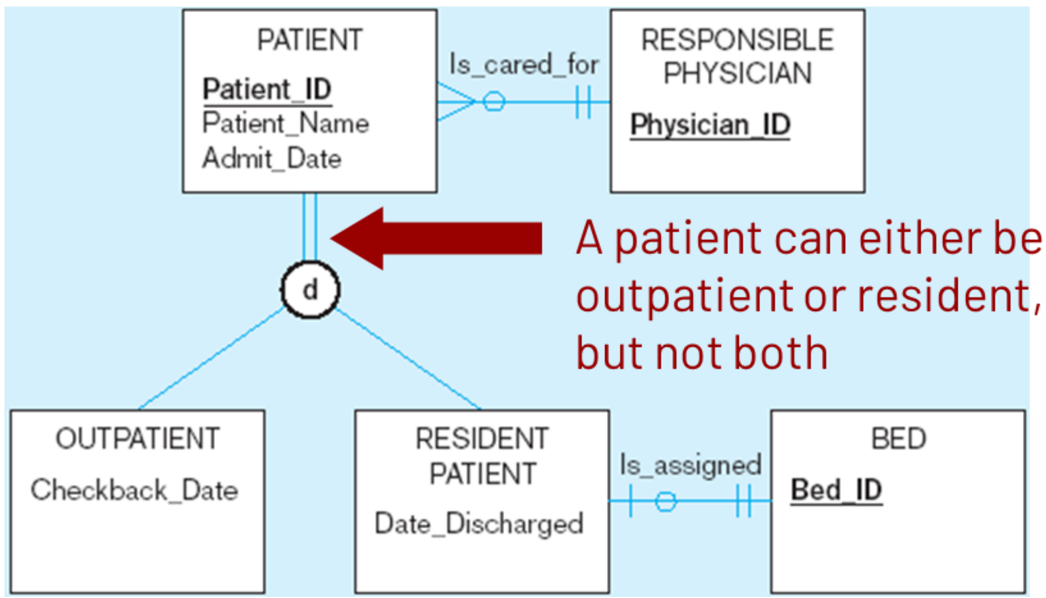
A vehicle could be a car, a truck, or neither

Constraints in Supertype/ Disjointness constraint

- **Disjointness Constraints**: Whether an instance of a supertype may simultaneously be a member of two (or more) subtypes
  - **Disjoint Rule**: An instance of the supertype can be only ONE of the subtypes
  - **Overlap Rule**: An instance of the supertype could be more than one of the subtypes

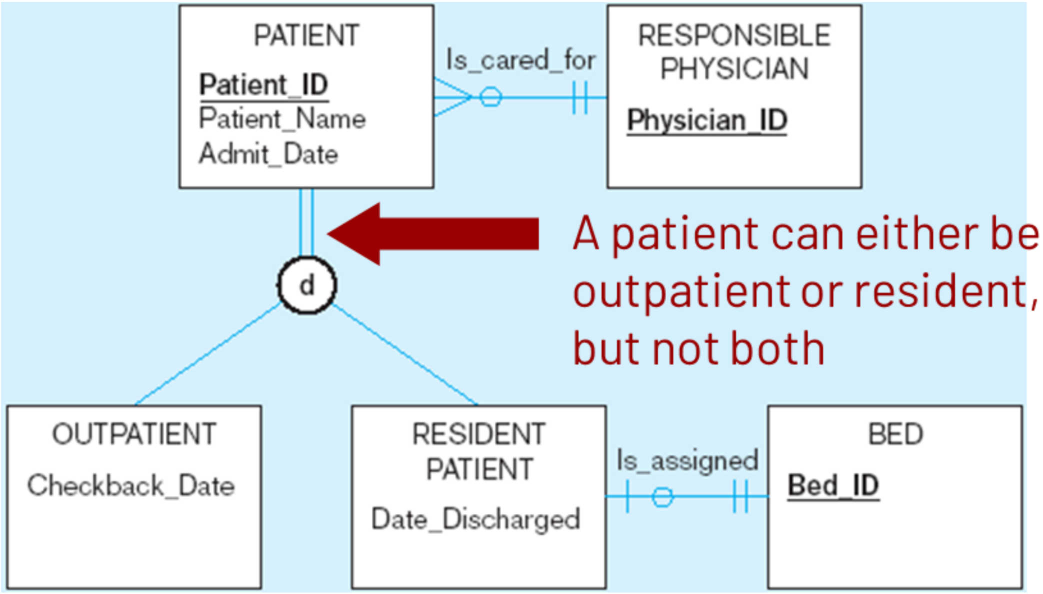
Constraints in Supertype/ Disjointness Constraint

- Examples of disjointness constraints
  - Disjoint Rule



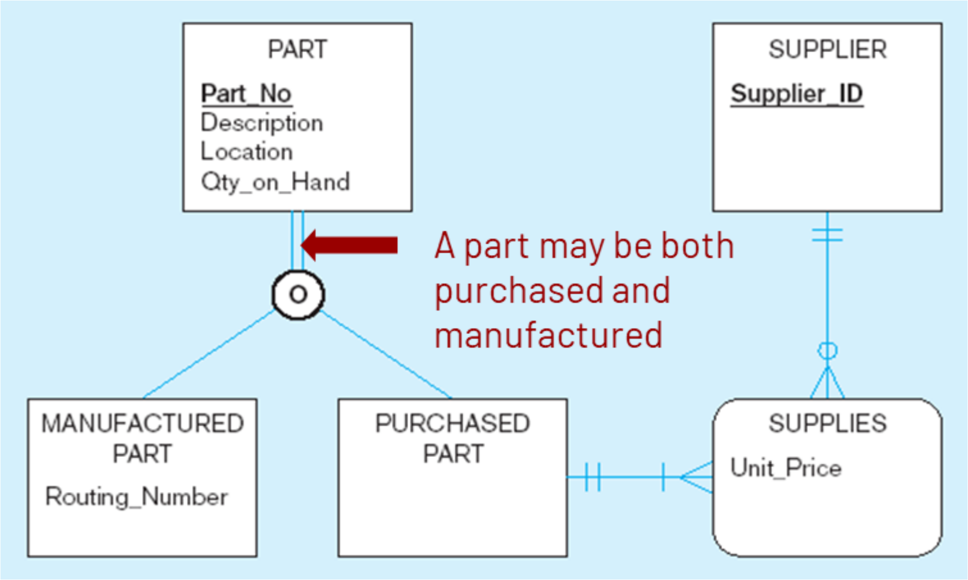
Constraints in Supertype/ Disjointness Constraint

- Examples of disjointness constraints
  - Disjoint Rule



Constraints in Supertype/ Disjointness Constraint

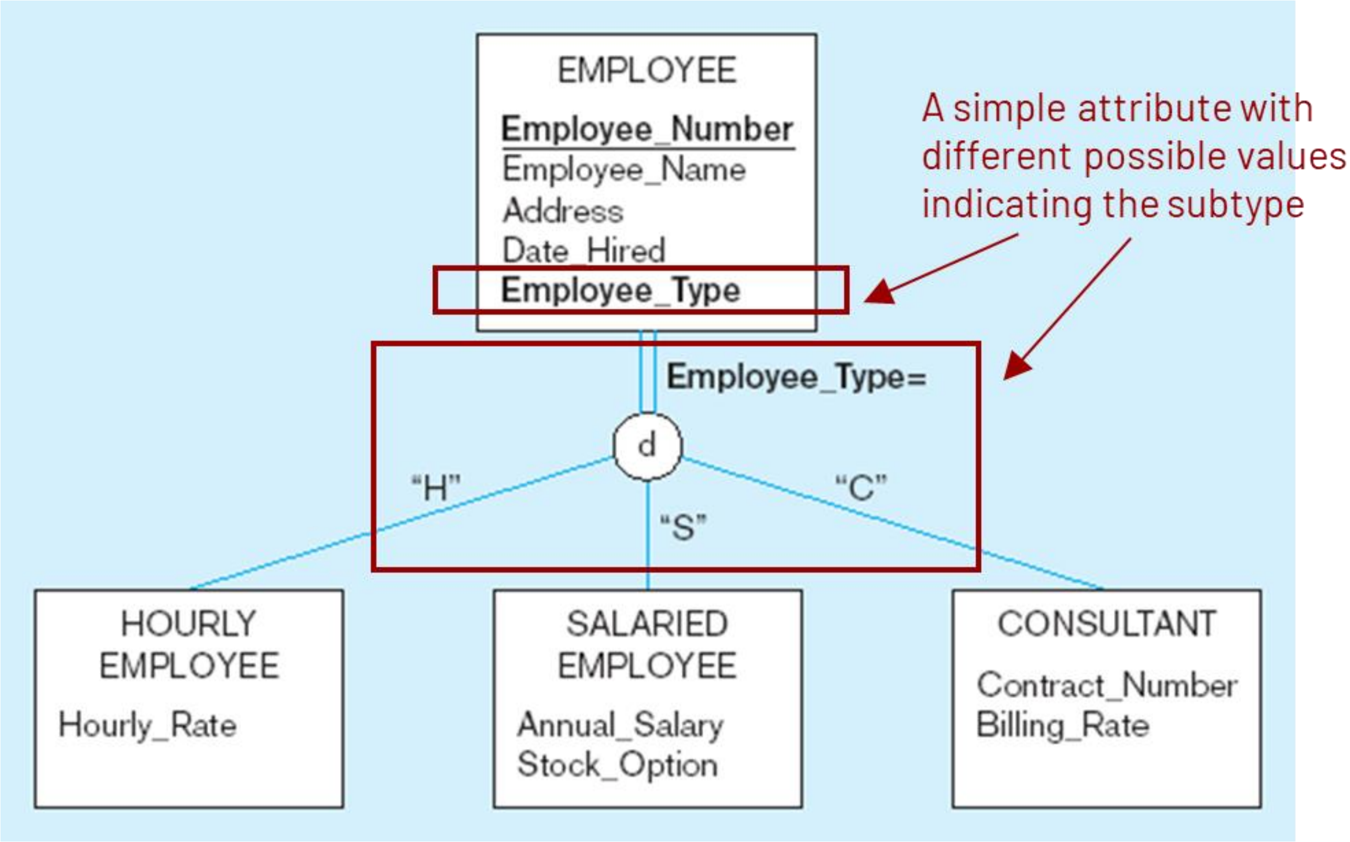
- Examples of disjointness constraints
  - Overlap Rule



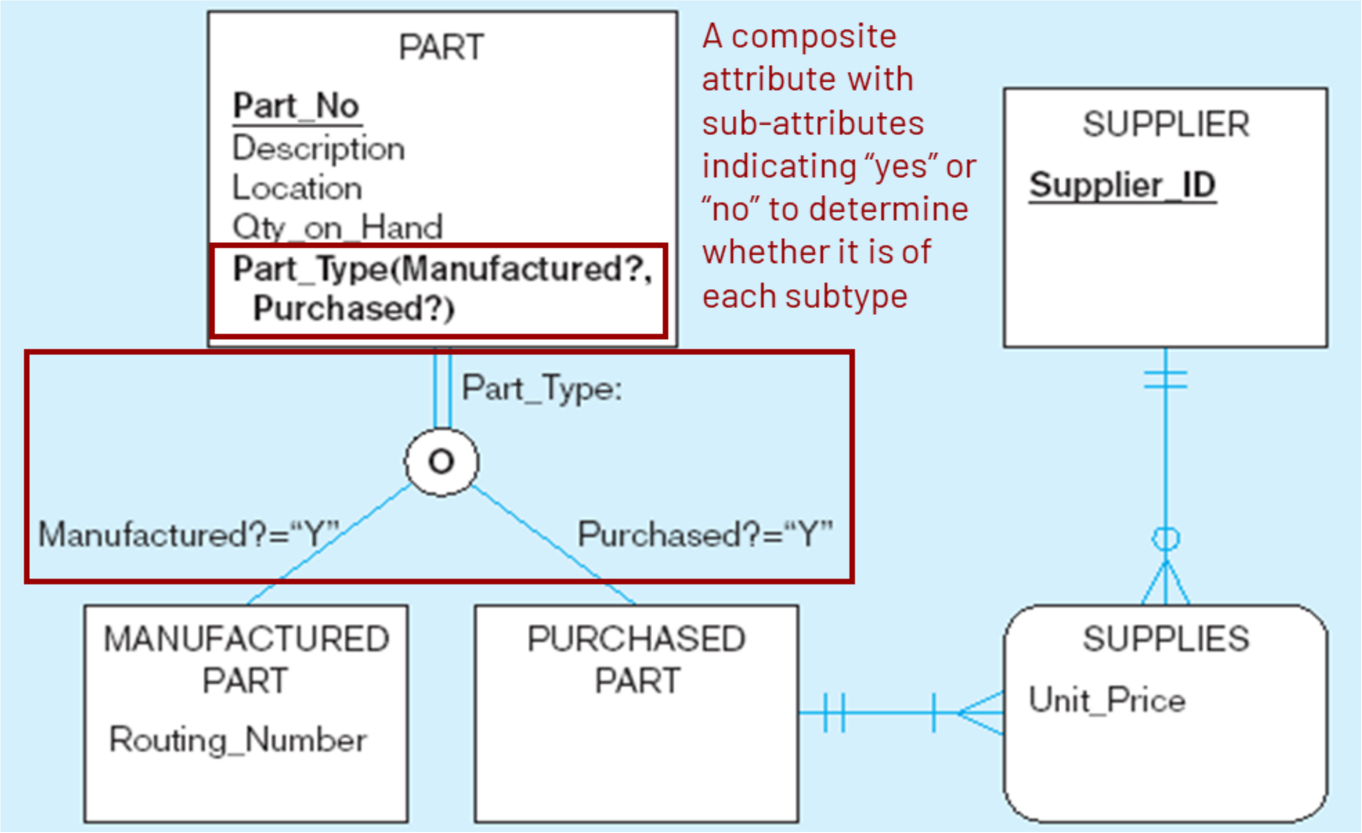
Constraints in Supertype/ Subtype Discriminators

- **Subtype Discriminator:** An attribute of the supertype whose values determine the target subtype(s)
  - **Disjoint** – a simple attribute with alternative values to indicate the possible subtypes
  - **Overlapping** – a composite attribute whose subparts pertain to different subtypes. Each subpart contains a boolean value to indicate whether or not the instance belongs to the associated subtype

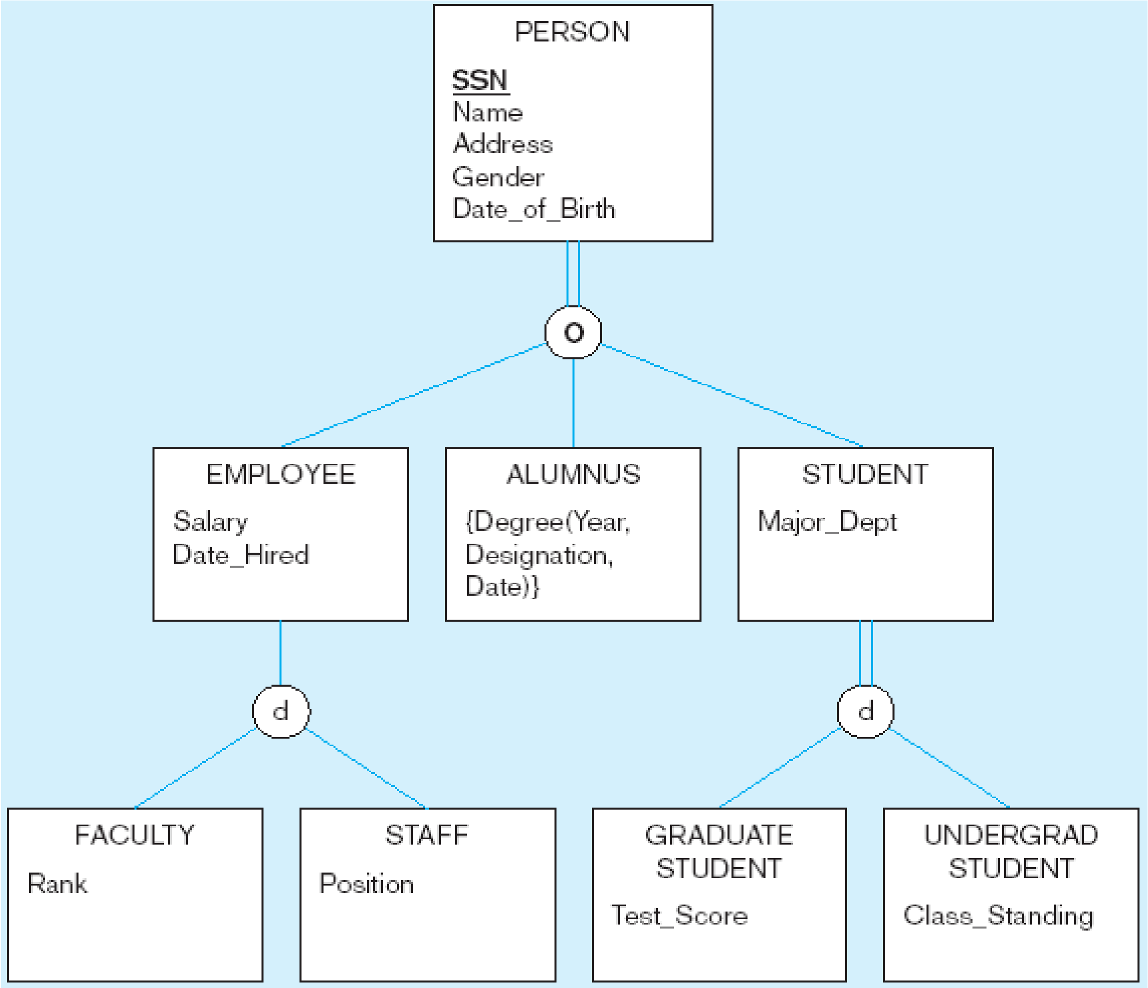
Constraints in Supertype/ Subtype Discriminators
(DISJOINT RULE)



Constraints in Supertype/ Subtype Discriminators
(OVERLAP RULE)



Example of Supertype/Subtype Hierarchy



NORMALIZATION OF DATABASE MODELS

Data Normalization

- Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that **avoid unnecessary duplication of data**
- The process of decomposing relations with anomalies to produce smaller, **well-structured** relations

Well-Structured Relations

- A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies
- Goal is to avoid anomalies
  - Insertion Anomaly–adding new rows forces user to create duplicate data
  - Deletion Anomaly–deleting rows may cause a loss of data that would be needed for other future rows
  - Modification Anomaly–changing data in a row forces changes to other rows because of duplication

General rule of thumb: A table should not pertain to more than one entity type

Example

EMPLOYEE

Emp_ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/200X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/200X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/200X
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/200X
110	Chris Lucero	Info Systems	43,000	C++	4/22/200X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/200X
150	Susan Martin	Marketing	42,000	Java	8/12/200X

Question–Is this a relation?

Answer–Yes: Unique rows and no multivalued attributes

Question–What's the primary key?

Answer–Composite: Emp_ID, Course_Title

Anomalies in Table Employee

- **Insertion**–can’t enter a new employee without having the employee take a class
- **Deletion**–if we remove employee 140, we lose information about the existence of a Tax Acc class
- **Modification**–giving a salary increase to employee 100 forces us to update multiple records

Why do these anomalies exist?
Because there are two themes (entity types) in this one relation. This results in data duplication and an unnecessary dependency between the entities

Anomalies in Table Employee

- **Insertion**–can't enter a new employee without having the employee take a class
- **Deletion**–if we remove employee 140, we lose information about the existence of a Tax Acc class
- **Modification**–giving a salary increase to employee 100 forces us to update multiple records

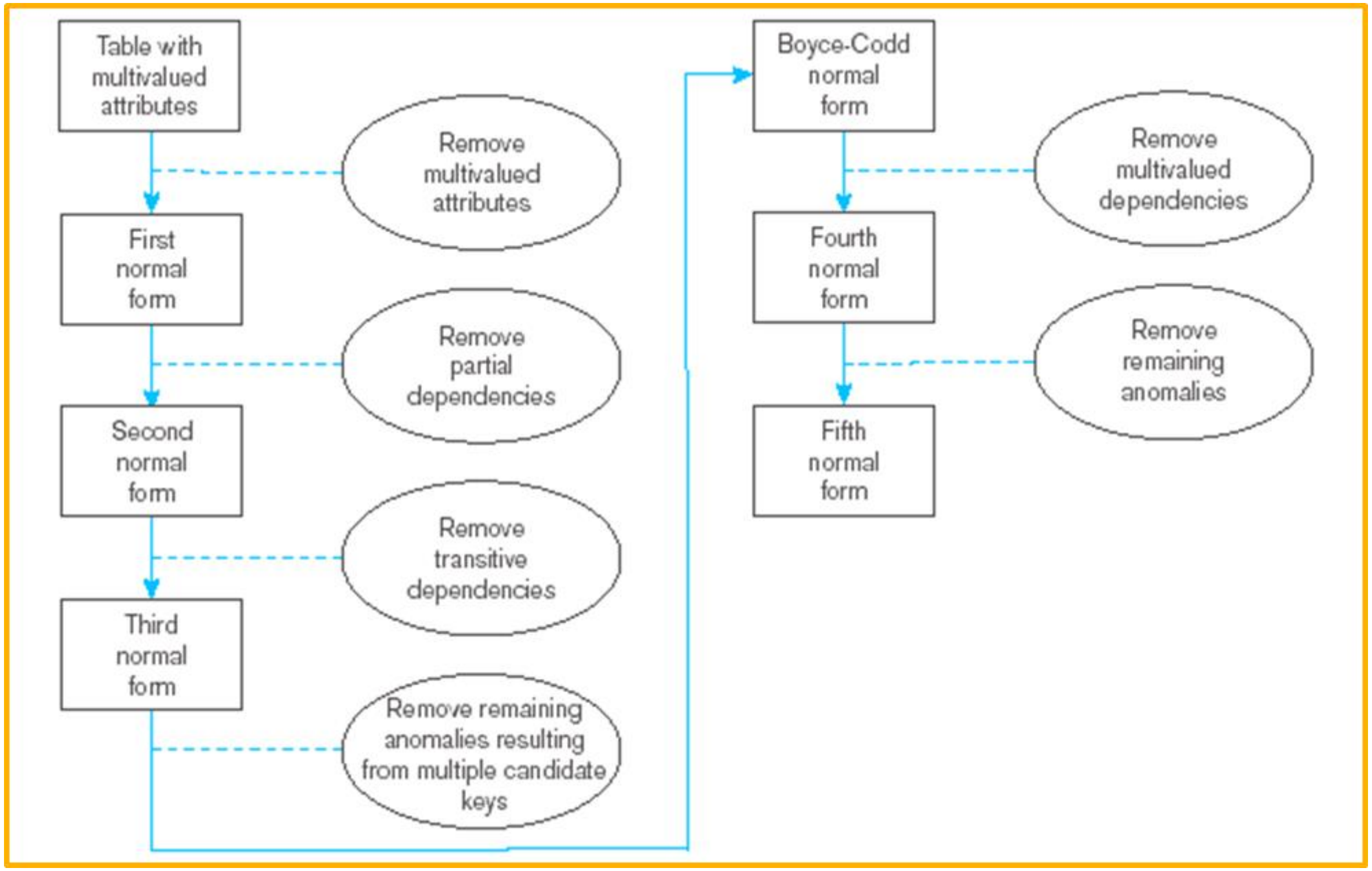
Why do these anomalies exist?

Because there are two themes (entity types) in this one relation. This results in data duplication and an unnecessary dependency between the entities

Anomalies in Table Employee

- Functional Dependency
  - The value of one attribute (the determinant) determines the value of another attribute
- Candidate Key
  - A unique identifier. One of the candidate keys will become the primary key
    - E.g. perhaps there is both credit card number and SS# in a table…in this case both are candidate keys
  - Each non-key field is functionally dependent on every candidate key

Steps in Normalization



First Normal Form

- No multivalued attributes
- Every attribute value is atomic
- **All relations are in 1st Normal Form**

Table with multivalued attributes, not in 1st normal form

Invoice Data

Order_ID	Order_ Date	Customer_ ID	Customer_ Name	Customer_ Address	Product_ID	Product_ Description	Product_ Finish	Unit_ Price	Ordered_ Quantity
1006	10/24/2006	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2006	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

Note: this is NOT a relation

Table with no multivalued attributes and unique rows, in 1st normal form

Invoice Relation

Product_ID → Product_Description, Product_Finish, Unit_Price
Order_ID, Product_ID → Ordered_Quantity

Order_ID	Order_ Date	Customer_ ID	Customer_ Name	Customer_ Address	Product_ID	Product_ Description	Product_ Finish	Unit_ Price	Ordered_ Quantity
1006	10/24/2006	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2006	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2006	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2006	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2006	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

Note: this is relation, but not a well-structured one

Table with no multivalued attributes and unique rows, in 1st normal form

Invoice Relation

Product_ID → Product_Description, Product_Finish, Unit_Price
Order_ID, Product_ID → Ordered_Quantity

Order_ID	Order_Date	Customer_ID	Customer_Name	Customer_Address	Product_ID	Product_Description	Product_Finish	Unit_Price	Ordered_Quantity
1006	10/24/2006	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2006	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2006	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2006	6	Furniture Gallery	Boulder, CO	11	4-Dr Dresser	Oak	500.00	4
1007	10/25/2006	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

Note: this is relation, but not a well-structured one

Anomalies in Invoice Table

Invoice Relation

Product_ID → Product_Description, Product_Finish, Unit_Price
Order_ID, Product_ID → Ordered_Quantity

Order_ID	Order_ Date	Customer_ ID	Customer_ Name	Customer_ Address	Product_ID	Product_ Description	Product_ Finish	Unit_ Price	Ordered_ Quantity
1006	10/24/2006	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2006	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2006	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2006	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2006	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

Note: this is relation, but not a well-structured one

- **Insertion**–if new product is ordered for order 1007 of existing customer, customer data must be re-entered, causing duplication
- **Deletion**–if we delete the Dining Table from Order 1006, we lose information concerning this item's finish and price
- **Update**–changing the price of product ID 4 requires update in several records

First Normal Form Example

Original Table

FirstName	LastName	Knowledge
Sheena	Aragon	Java, PHP, C++, Python
Jonalyn	Lupera	PHP, Java
Karen	Redonda	Java, C++, Cobol

First Normal Form

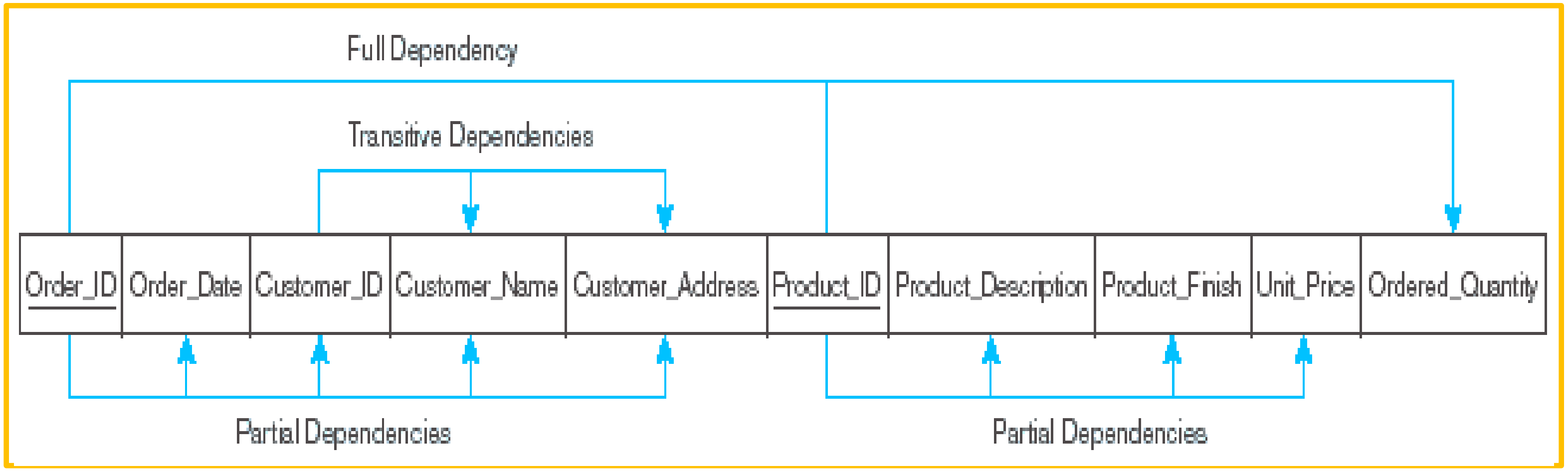
FirstName	LastName	Knowledge
Sheena	Aragon	Java
Sheena	Aragon	PHP
Sheena	Aragon	C++
Sheena	Aragon	Python
Jonalyn	Lupera	PHP
Jonalyn	Lupera	Java
Karen	Redonda	Java
Karen	Redonda	C++
Karen	Redonda	Cobol

NOTE: To get to the first normal form (1NF) we must create a separate tuple for each value of the multivalued attribute

Second Normal Form

- 1NF PLUS **every non-key attribute is fully functionally dependent on the ENTIRE primary key**
  - Every non-key attribute must be defined by the entire key, not by only part of the key
  - No partial functional dependencies

Functional dependency diagram for INVOICE



Order_ID → Order_Date, Customer_ID, Customer_Name, Customer_Address

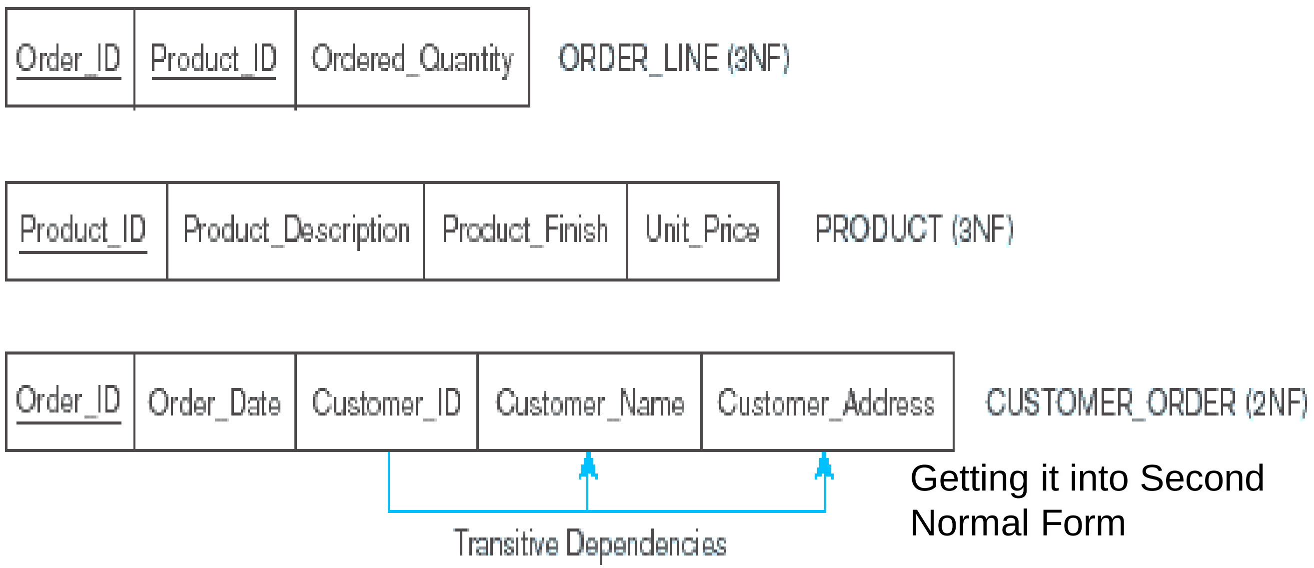
Customer_ID → Customer_Name, Customer_Address

Product_ID → Product_Description, Product_Finish, Unit_Price

Order_ID, Product_ID → Order_Quantity

Therefore, NOT in 2nd Normal Form

Removing Partial Dependencies



Getting it into Second Normal Form

Partial dependencies are removed, but there are still transitive dependencies

Second Normal Form Example

s_id	s_name	p_id	profName	grade
1902	Lico	101	Velasco	1.4
1903	Bayta	102	Sias	1.9
1904	Celaje	103	Benosa	1.5

The table in this example is in first normal form (1NF) since all attributes are single valued. But it is not yet in 2NF. If student **1902** leaves university and the tuple is deleted, then we loose all information about professor **Velasco**, since this attribute is fully functional dependent on the primary key **s_id**. To solve this problem, we must create a new table Professor with the attribute Professor (the name) and the key **p_id**. The third table Grade is necessary for combining the two relations Student and Professor and to manage the grades. Besides the grade it contains only the two IDs of the student and the professor. If now a student is deleted, we do not loose the information about the professor.

2nd Normal Form

s_id	s_name
1902	Lico
1903	Bayta
1904	Celaje

p_id	profName
101	Velasco
102	Sias
103	Benosa

s_id	p_id	grade
1902	101	1.4
1903	102	1.9
1904	103	1.5

Third Normal Form

- 2NF PLUS ***no transitive dependencies*** (functional dependencies on non-primary-key attributes)
- Note: This is called transitive, because the primary key is a determinant for another attribute, which in turn is a determinant for a third
- Solution: Non-key determinant with transitive dependencies go into a new table; non-key determinant becomes primary key in the new table and stays as foreign key in the old table

Removing Partial Dependencies

BEFORE

Order_ID	Order_Date	Customer_ID	Customer_Name	Customer_Address

AFTER

Order_ID	Order_Date	Customer_ID

ORDER (3NF)

Customer_ID	Customer_Name	Customer_Address

CUSTOMER (3NF)

Transitive dependencies are removed

Third Normal Form Example

Original Table/1st/2nd Normal Form

v_id	v_name	account_num	bank_code	bank

3rd Normal Form

v_id	v_name	account_num	bank_code

bank_code	bank

The table in this example is in 1NF and in 2NF. But there is a transitive dependency between bank_code and bank, because bank_code is not the primary key of this relation. To get to the third normal form (3NF), we have to put the bank name in a separate table together with the clearing number to identify it.