DATABASE - 1

3RD CLASS

COMPUTER SCIENCE DEPARTMENT

5th Lecture – Mapping Relationships

Sunday 20th of Oct. 2024

LECTURER :

DR. RAYAN YOUSIF ALKHAYAT

STEP 3 : Map Binary Relationships :

- The procedure for representing relationships depends on both
- 1. The degree of the relationships (unary, binary and ternary)
- And
- 2. The cardinalities of the relationships.

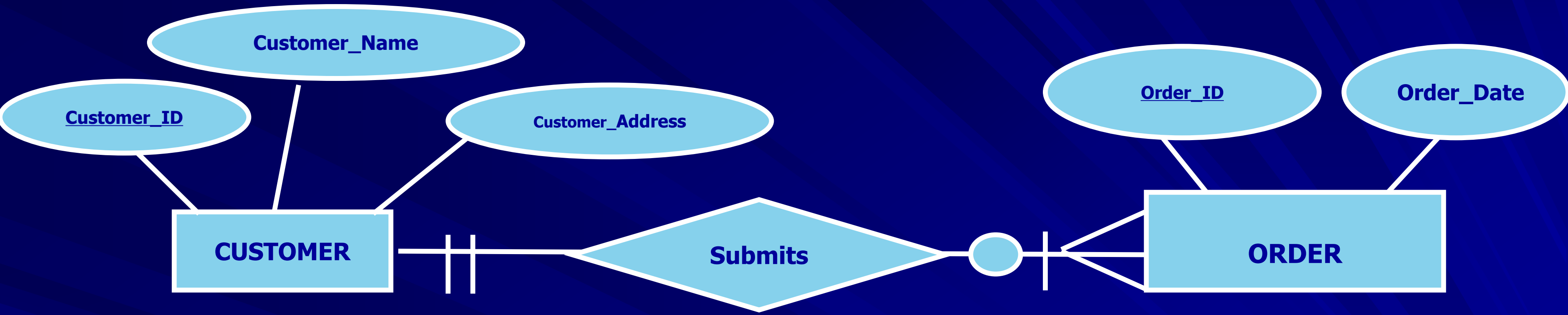
A : Map Binary One-to-Many Relationships :

1. **For each binary 1:M relationship create a relation for each of the two entity types participating in the relationship using the procedure in Step 1.**

2. **Include the primary key attribute (or attributes) of the entity on the one-side of the relationship as a foreign key in the relation that is one to many – side of the relationship (to remember: The primary key migrates to the many side)**

We illustrate the above rules using the relationships between CUSTOMER and ORDER, which is 1:M relationship, in the figure bellow. The primary key Customer_ID of CUSTOMER (the one side) is included as a foreign key in ORDER(the many -side) .The foreign key relationship is indicated with an arrow.

Part (a)



Part (b)

CUSTOMER

<u>Customer_Id</u>	Customer_Name	Customer_Address

ORDER

<u>Order_ID</u>	Order_Date	<u>Customer_Id</u>

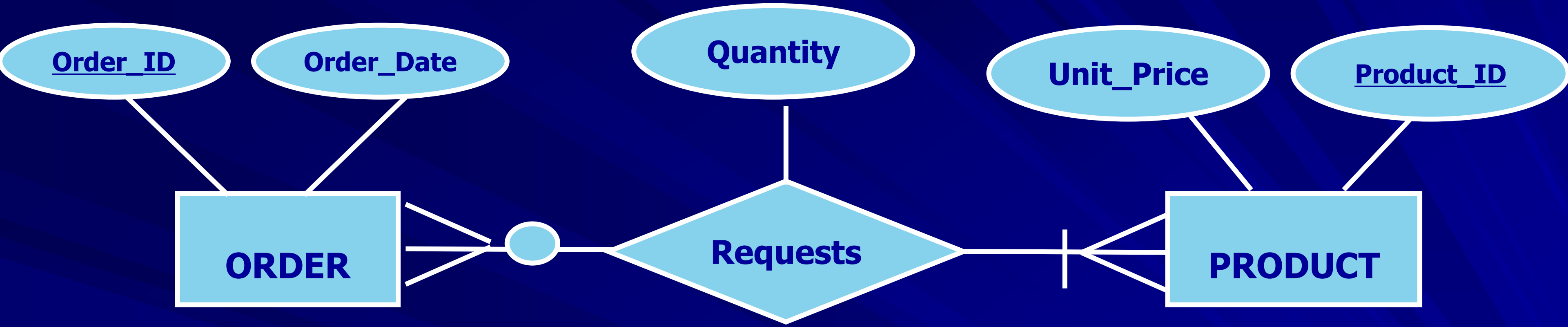
B: Map Binary Many –to Many Relations:

- Supposing we have binary (M: N) relationship between two entity types A and B.
- For such relationship we create a new relation C. Includes as foreign key attributes in C the primary key for each of the two participating entity types.
- These attributes become the primary key of C.
- Any non-key attributes are associated with M: N relationships are included with the relation C.

- An example of applying this rule is in the figure below:

- Part (a) shows the requests relationship between entity types ORDER and PRODUCT
- Part(b) shows three relations (ORDER, ORDER LINE and PRODUCT) that are formed from the entity types and requests relationships.
- First, a relation created for each of the two regular entity types ORDER, ORDER LINE and PRODUCT.
- Then a relation (named ORDER_LINE is created for the requests relationship.
- The primary key of ORDER_LINE is the combination of Order_ID and Product_ID, which are the respective primary keys of ORDER and PRODUCT .
- As indicated in the diagram, these attributes are foreign keys that “point to “the respective keys. The non-key attribute Quantity also appears in ORDER_LINE.

Part (a)



Part (b)

ORDER

Order_Id	Order_Date

ORDER LINE

Order_Id	Product_Id	Quantity

PRODUCT

Product_Id	Unit_Price	Other Attributes

C: Map Binary One-to-One Relationships:

Binary one- to -one relationships can be viewed as a special case of one- to many relationships .

The process of mapping such a relationship to relation requests two steps :

1. First, two relations are created , one for each of the participating entity types .
2. Second , the primary key of one of the relations is included as a foreign key in the other relation.

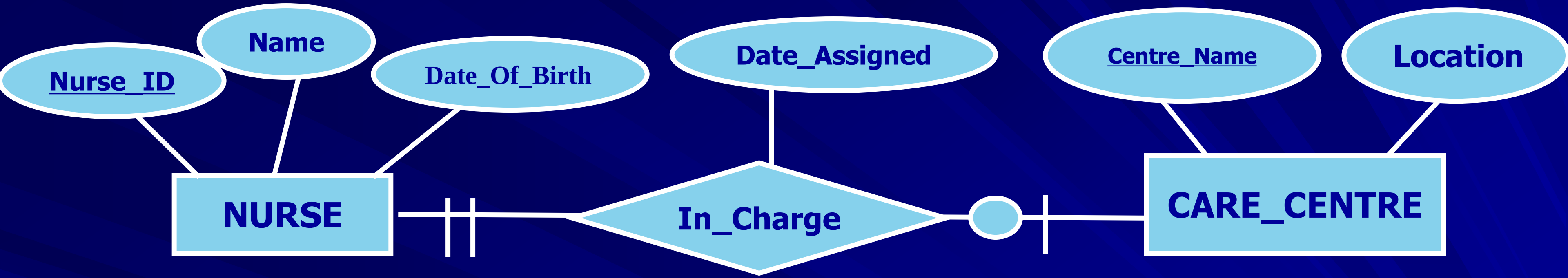
- In 1: 1 relationship, the association in one the one direction is nearly always optional one , while the association in the other direction is mandatory one .
- The foreign key of the entity type that has a mandatory participation in the 1:1 relationship should be included in the relation on the optional side of the relationship.
- This approach will avoid the need to store null values in the foreign key attribute .
- Any attributes associated with the relationship itself are also included in the same relation as the foreign key.

- As an example of applying the procedure is shown below, part (a) shows a binary 1:1 relationship between the entity type NURSE and CARE_CENTRE.
- Each care centre must have a nurse who is in charge of that centre.
- Thus the association from CARE_CENTRE to NURSE is a mandatory one, while the association from NURSE to CARE_CENTRE is an optional one(since any nurse may or may not be in charge of a care centre).
- The attribute Date_Assigned is attached to the In_Charge relationship.

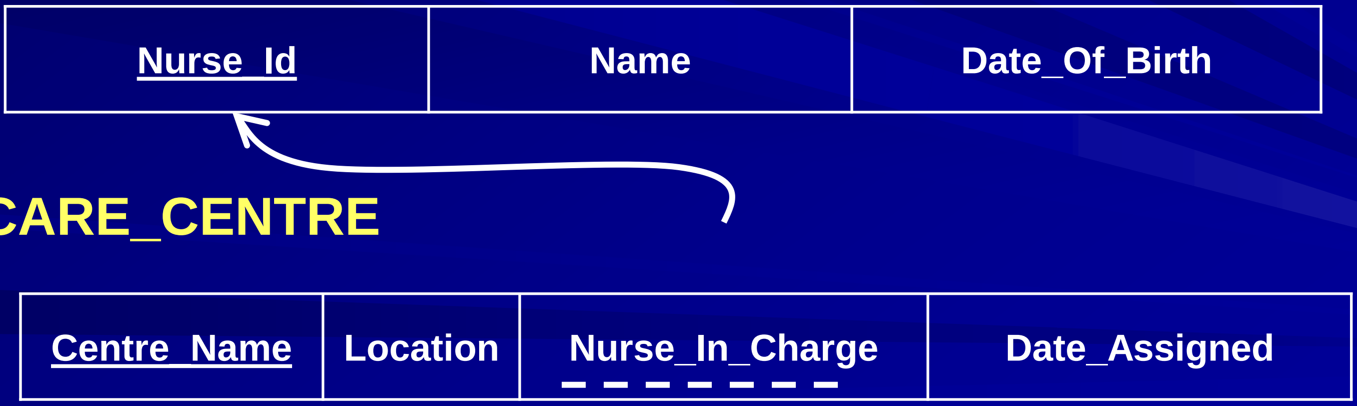
.

- The result of mapping this relationship to a set of relations is shown in part (b). The two relations NURSE and CARE_CENTRE are created from the two entity types.
- Since CARE_CENTRE is the optional participant, the foreign key is placed in this relation.
- In this case, the foreign key called Nurse_in_Charge. It has the same domain as Nurse_ID , and the relationship with the primary key is shown in the figure.
- The attribute Date_assigned is also located in CARE_CENTRE.

Part (a)



Part (b)



NURSE

Nurse_Id	Name	Date_Of_Birth

CARE_CENTRE

Centre_Name	Location	Nurse_In_Charge	Date_Assigned

Step 4: Map Associative Entities:

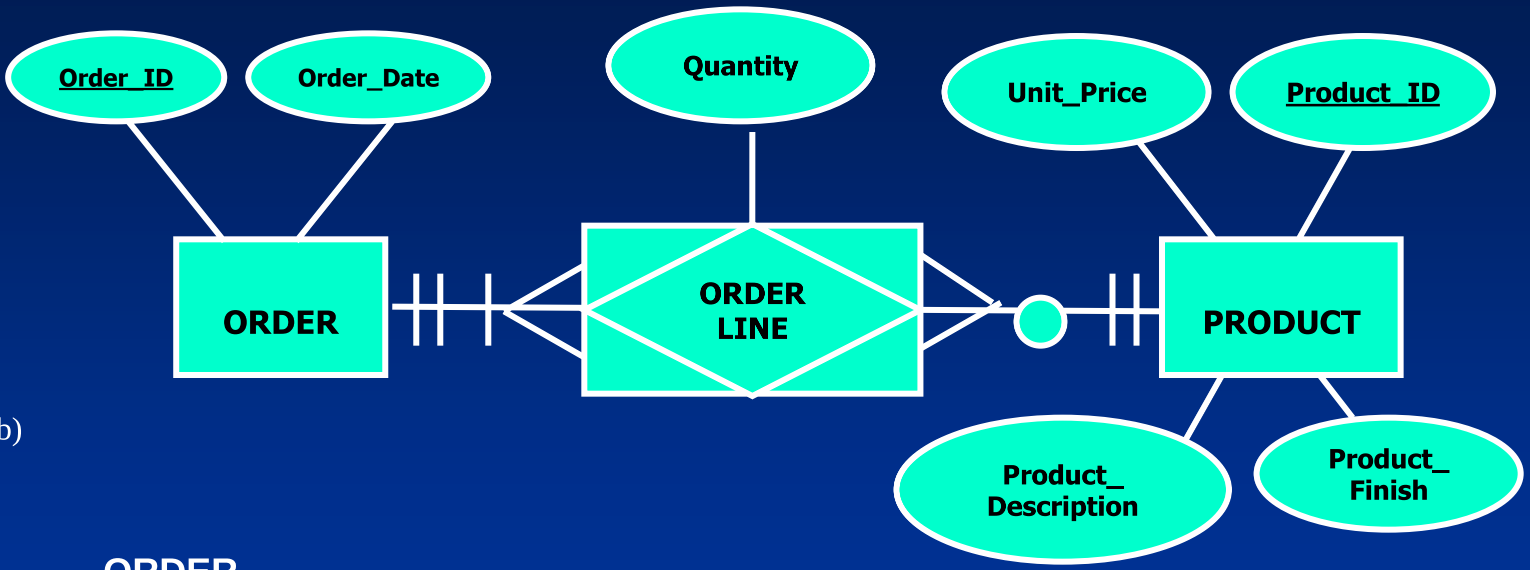
- It is a good approach when the end user can best visualize the relationship as an entity type rather than as an M:N relationship.

1. Create three relations: one for each of the participating entity types ,and the third for the associative entity. We refer to the relation formed from the associative entity as the *associative relation*.
2. Depending on weather the on the E-R diagram an identifier was assigned to the associative entity.

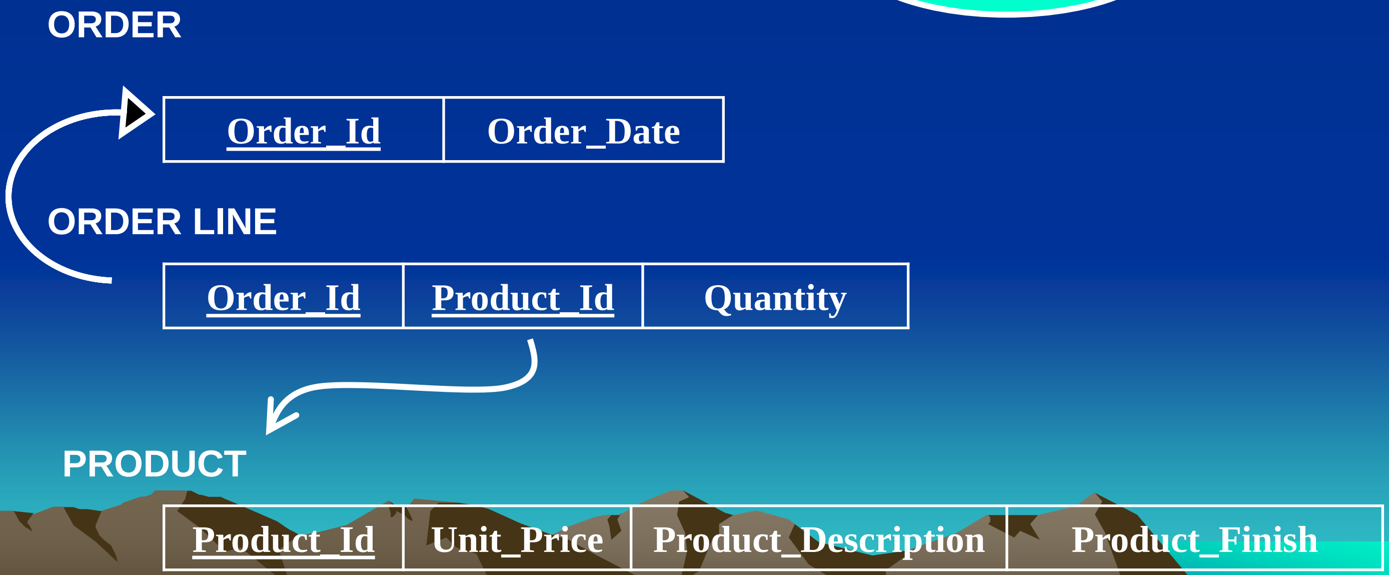
CASE A: Identifier Not Assigned:

- If an identifier was not assigned, the default primary key for the associative relation consists of the two primary key attributes from the other two relations. These attributes are then foreign keys that reference the other two relations.
- An example of this case is shown in figure below ,part(a)shows the associative entity ORDER_LINE that links the ORDER and PRODUCT entity types, part(b) shows the three relations that result from this mapping.

Part (a)



Part (b)



ORDER

<u>Order_Id</u>	Order_Date

ORDER LINE

<u>Order_Id</u>	<u>Product_Id</u>	Quantity

PRODUCT

<u>Product_Id</u>	Unit_Price	Product_Description	Product_Finish

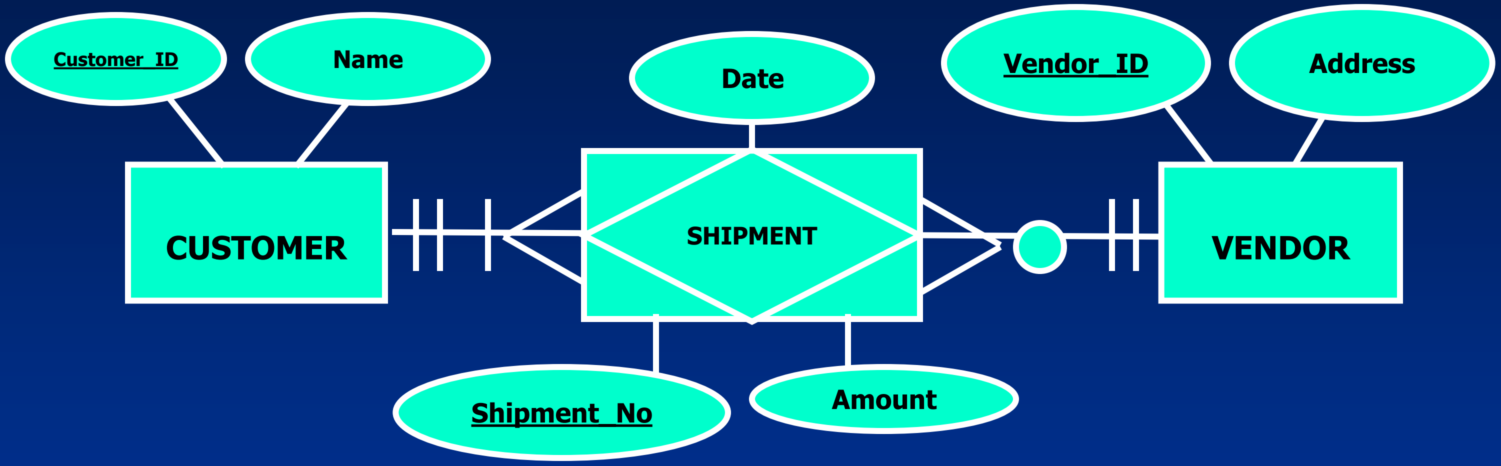
CASE B: Identifier Assigned:

- some times the data modeler will assign an identifier (called *surrogate* identifier or key) to the associative entity type on the E-R Diagram .There are two reasons that may motivate this approach :

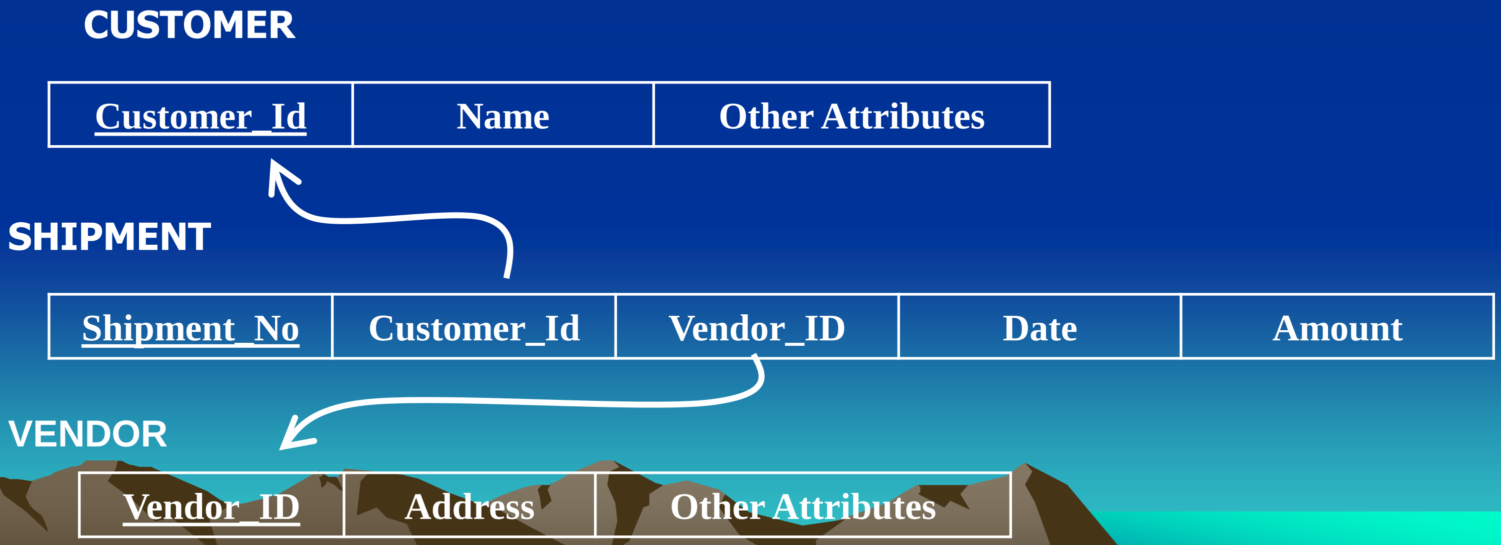
1. The associative entity type has a natural identifier that is familiar to the end users.
2. The default identifier (consisting of the identifiers for each of the participating entity types) may not uniquely identify instances of the associative entity.

- The process for mapping the associative entity in this case is now modified as follows.
- As before, a new (associative) relation is created to represent the associative entity. However, the primary key for this relation is the identifier assigned on E-R diagram (rather than the default key).
- The primary key for the two participating entity types are included as foreign keys in the associative relation.
- An example of this process is in the figure below, part (a) shows the associative entity type SHIPMENT that links the CUSTOMER and VENDOR entity types .Shipment_No has been chosen as the identifier for SHIPMENT, for two reasons

Part (a)



Part (b)



CUSTOMER

Customer_Id	Name	Other Attributes

SHIPMENT

Shipment_No	Customer_Id	Vendor_ID	Date	Amount

VENDOR

Vendor_ID	Address	Other Attributes

Shipment_No is a natural identifier for this entity type that is familiarto end users.

The default identifier consisting of the combination of Customer_ID and Vendor_Id does not uniquely identify the instances of SHIPMENT . In fact, a given vendor will make many shipments to a given customer .

Even including the attribute date does not guarantee uniqueness, since there may be more than one shipment by particular vendor on a given date , but the surrogate key Shipment_No will uniquely identify each shipment.

Tow non-key attributes associated with SHIPMENT are Date and Amount .

The result of mapping this entity to a set of relations is in part (b).

The new associative relation is named SHIPMENT.

The primary key is Shipment_No. Customer_ID and Vendor_ID are included as foreign keys in this relation, and Date and Amount are Non-Key attributes.

Step 5: Map Unary Relationships:

- Unary relationships = recursive relationships

- Two most important unary relationship cases are: one- to –many

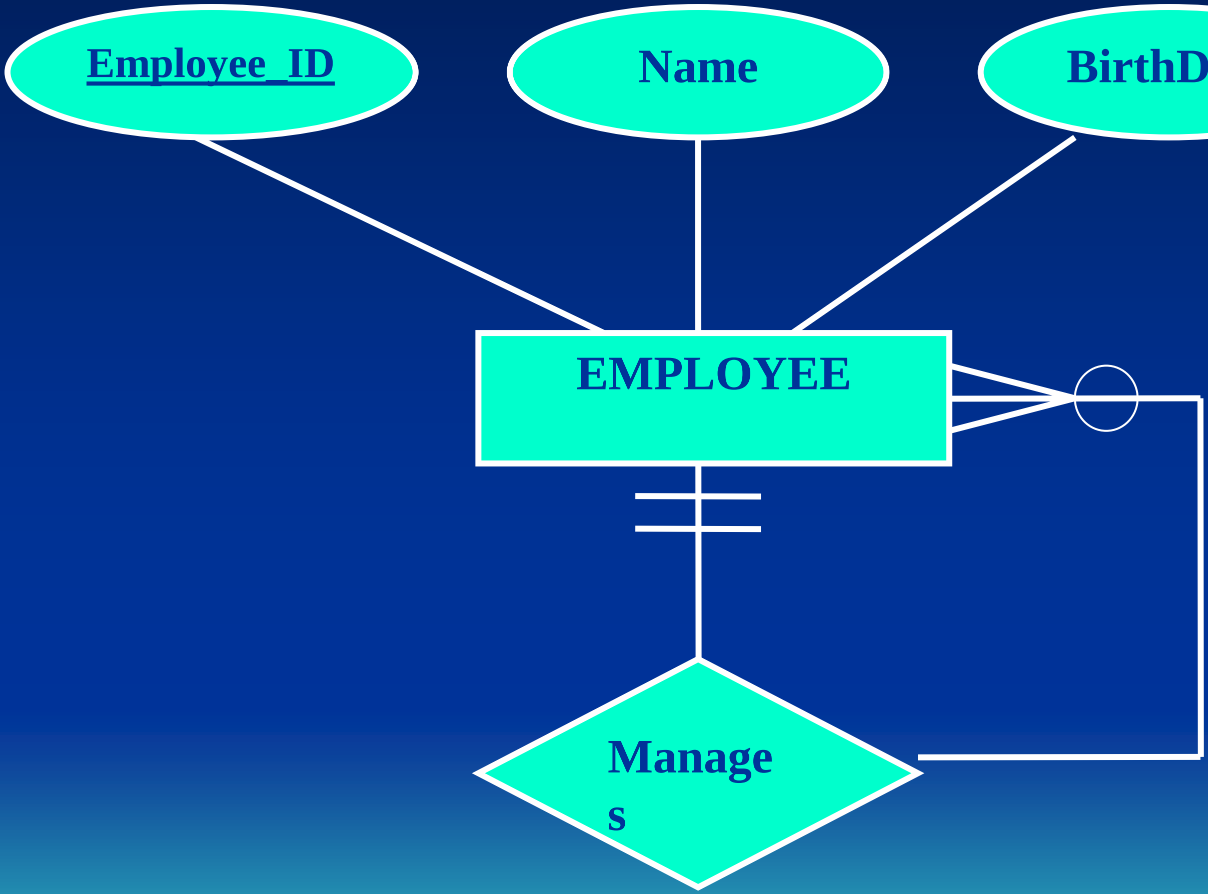
  and many –to – many.

1. Unary one- to – many relationships:

- The entity type in the unary relationship mapped to a relation using the procedure in step (1).
- Then a foreign key attribute added *within* the same relation that references the primary key values (this foreign key must have the same domain as the primary key).
- A recursive foreign key is a foreign key in a relation that references the primary key values of the same relation .

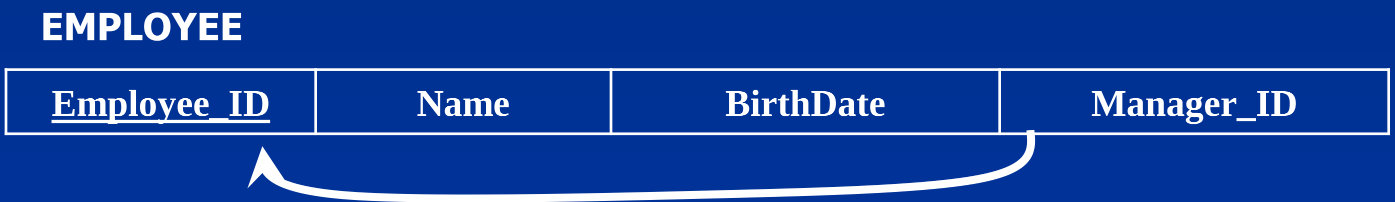
- The figure below shows in part(a) a unary one –to-many relationship named Manages that associate each employee of an organization with another employee who is his or her manager. Each employee has exactly one manager; a given employee may manage zero to many employees .

Part (a)



- The EMPLOYEE relation that results from mapping this entity and relationship is shown in part(b) .
- The (recursive) foreign key in the relation named Manager_ID. This attribute has the same domain as the primary key Employee_ID.
- Each row of this relation stores the following data

Part (b)



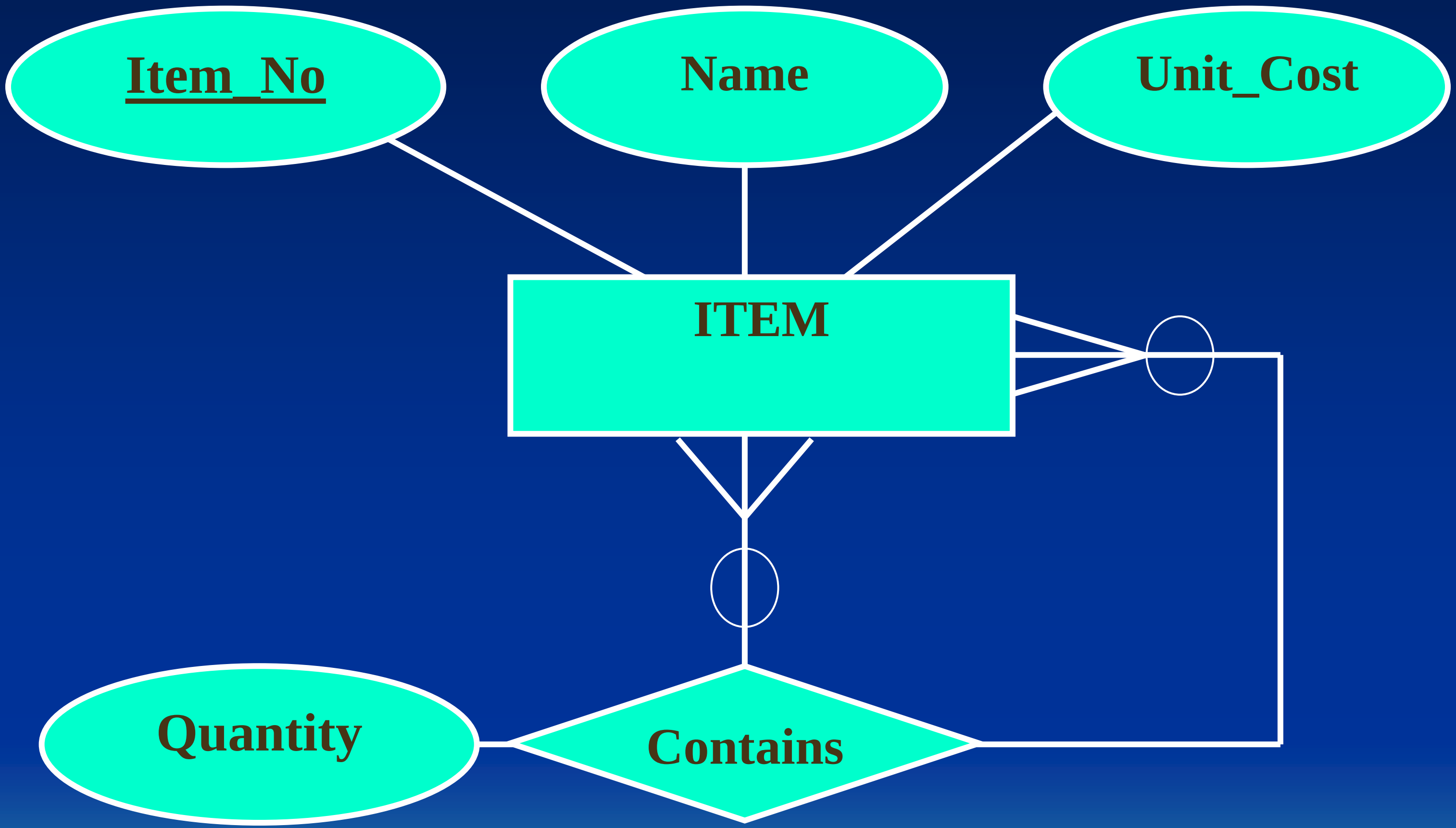
2:Unary many-to many relationships:

- In this type two relations are created: one to represent the entity type in the relation and the other an associative relation to represent the M: N relationship itself.
- The primary key of the associative relation consists of two attributes .
- These attributes (Which need not to have the same name) both take their values from the primary key of the other relation.
- Any non-key attribute of the relationship is included in the associative relation.

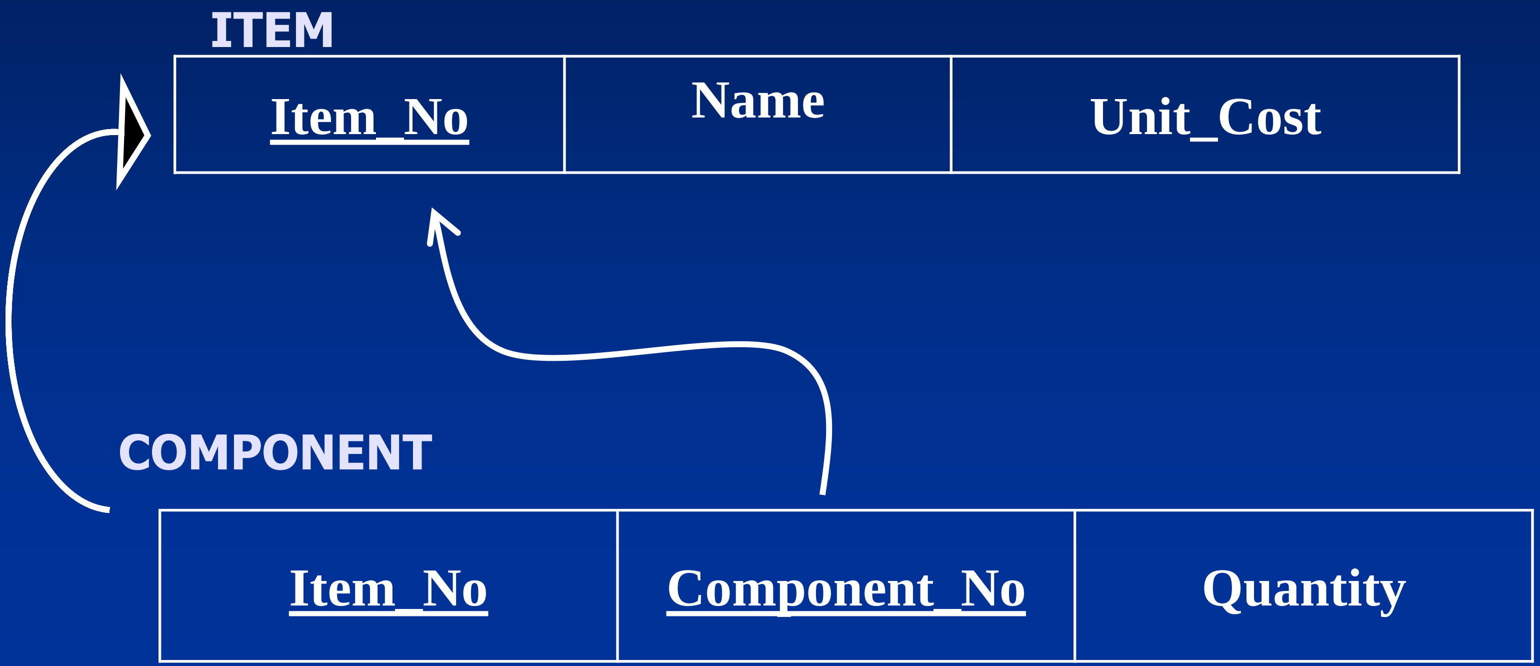
2:Unary many-to many relationships:

- An example of mapping a unary M:N relationship is shown in figure below, part(a) shows a bill -materials among items that are assembled from other items or components, the relationship (called Contains) is M:N since a given item can contain numerous component items, and conversely an item can be used as a component in numerous other items.

Part (a)



Part (b)


ITEM
Item_No
Name
Unit_Cost
COMPONENT
Item_No
Component_No
Quantity

Step 6: Map Ternary (and *n*-ary) Relationships:

Recall that a ternary relationship is a relationship among three entity types.

In fact it is recommended to convert a ternary relationship to an associative entity in order to represent participation constrains more accurately.

Step 6: Map Ternary (and *n*-ary) Relationships:

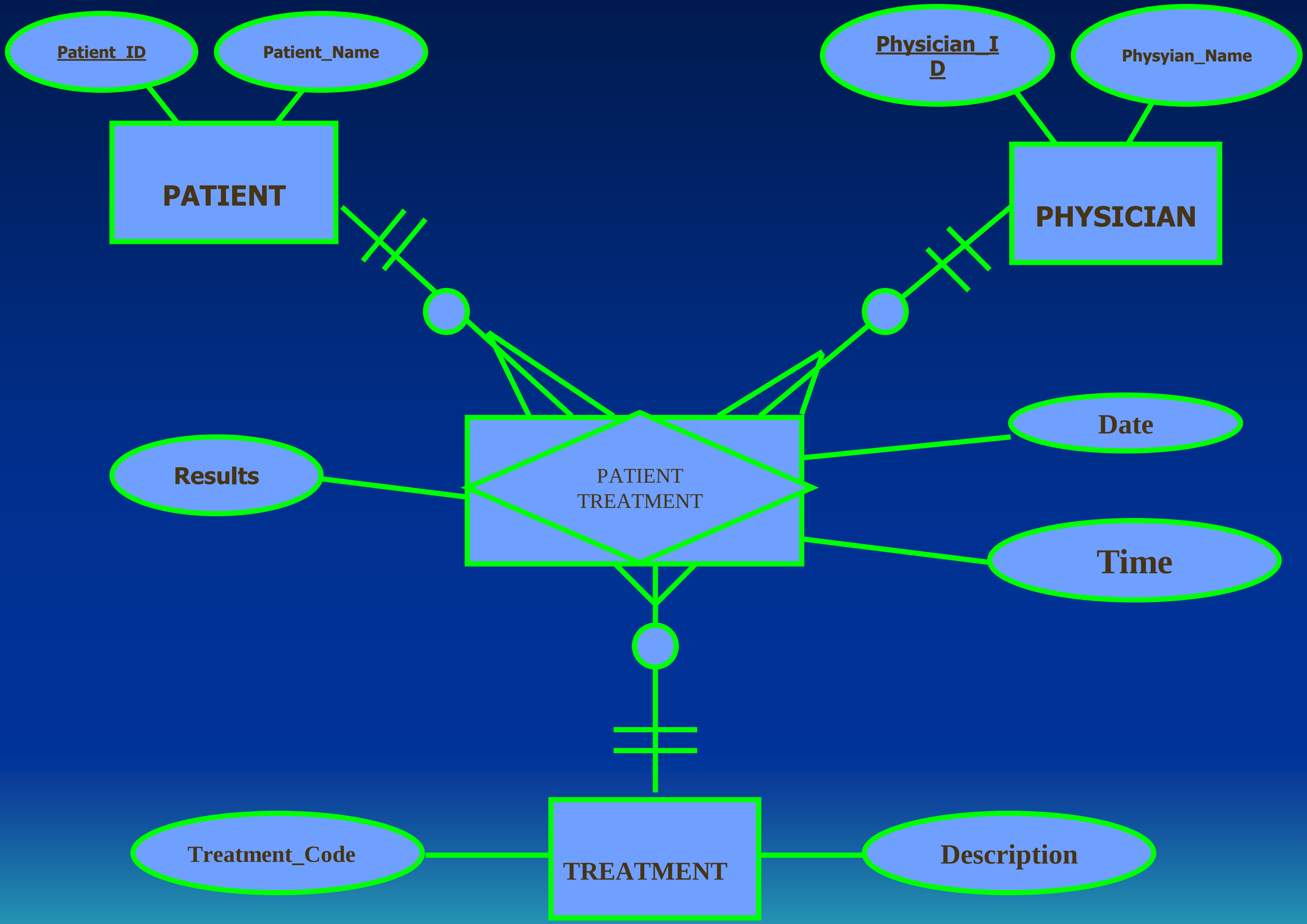
To map associative entity type that links three regular entity types or more we create a new associative relation the default primary key of this relation consists of the three primary key attributes for the participating entity types

(in some cases additional attributes are required to form a unique primary key) .

These attributes then act in the role of foreign keys that reference the individual primary keys of the participating entity types any attributes of the associative entity type become attributes of the new relation .

An example of mapping ternary relationship (represented as an associative entity type) is shown in figure below, in part(a) is an E-R segment (or view) that represents *patient* receiving a *treatment* from a *physician* the associative entity type PATIENT TREATMENT has the attributes Date, Time, and results ; values are recorded for these attributes for each instance of PATIENT TREATMENT .

Part (a)

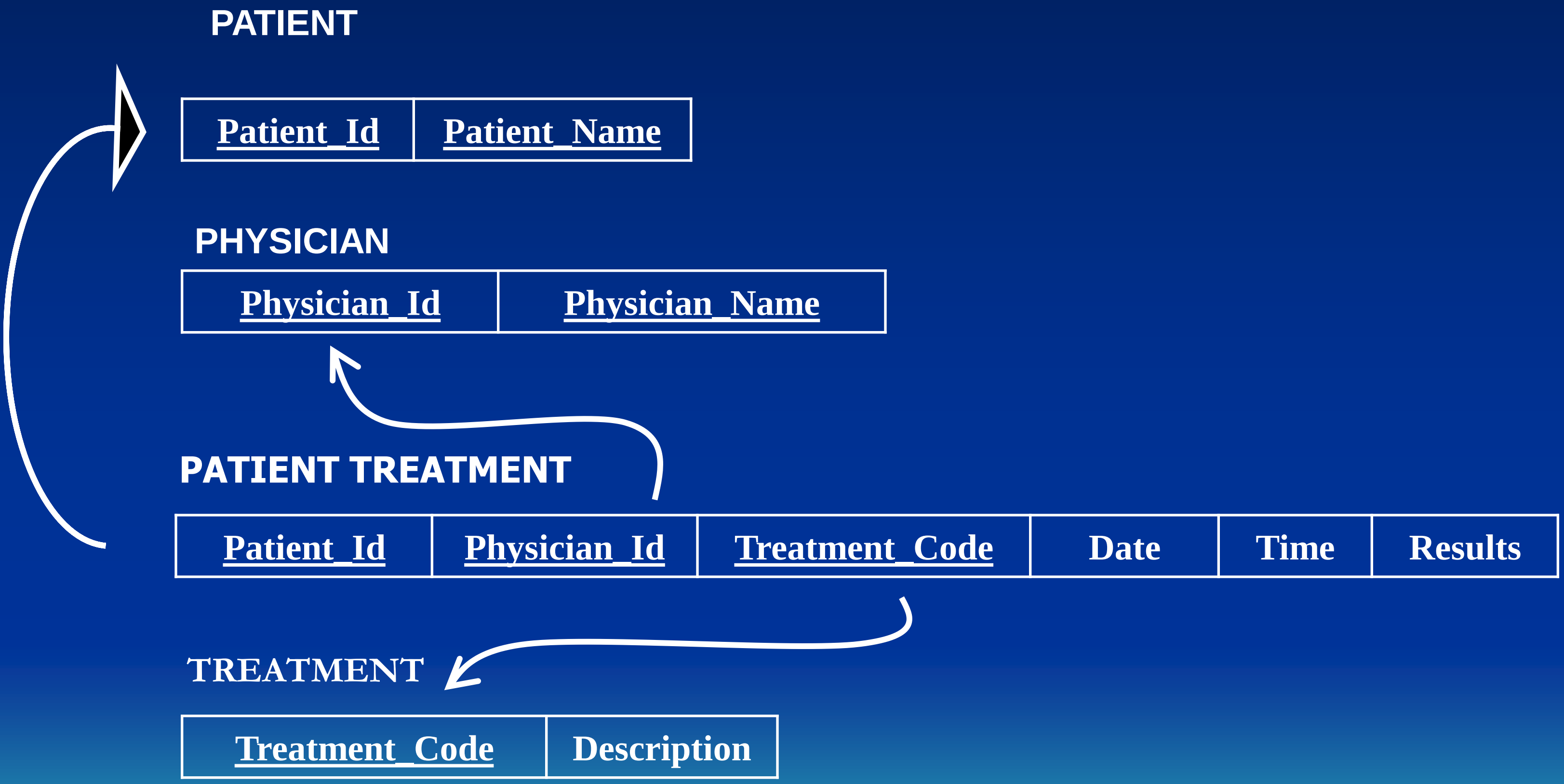

Patient_ID
Patient_Name
PATIENT
Physician_I D
Physyian_Name
PHYSICIAN
Results
PATIENT TREATMENT
Date
Time
Treatment_Code
TREATMENT
Description

- The result of mapping this view is shown in part (b) .
- The primary key attributes Patient_ID ,Physician_ID,and Treatment_Code become foreign keys in PATIENT TREATMENT. These attributes are components of the primary key of PATIENT TREATMENT.
- However they do not uniquely identify a given treatment since a patient may receive the same treatment from the same physician on more than one occasion .

- Does including the attribute Date as part of the primary key (along with the other three attributes) result in primary key ?
- This would be so if a given patient receives only one treatment from a particular physician on a given date.

- However, this is not likely to be the case. For example, a patient may be receiving a treatment in the morning , then the same treatment in the afternoon .
- To resolve this issue we include time as apart of the primary key.
- Therefore ,a primary key of PATIENT TREATMENT consists of the five attributes shown in part(b) : Patient_UD,Physician_ID, Treatment_Code, Date,and Time.
- The only non key attribute in the relation is Results.

Part (b)



PATIENT

Patient_Id	Patient_Name

PHYSICIAN

Physician_Id	Physician_Name

PATIENT TREATMENT

Patient_Id	Physician_Id	Treatment_Code	Date	Time	Results

TREATMENT

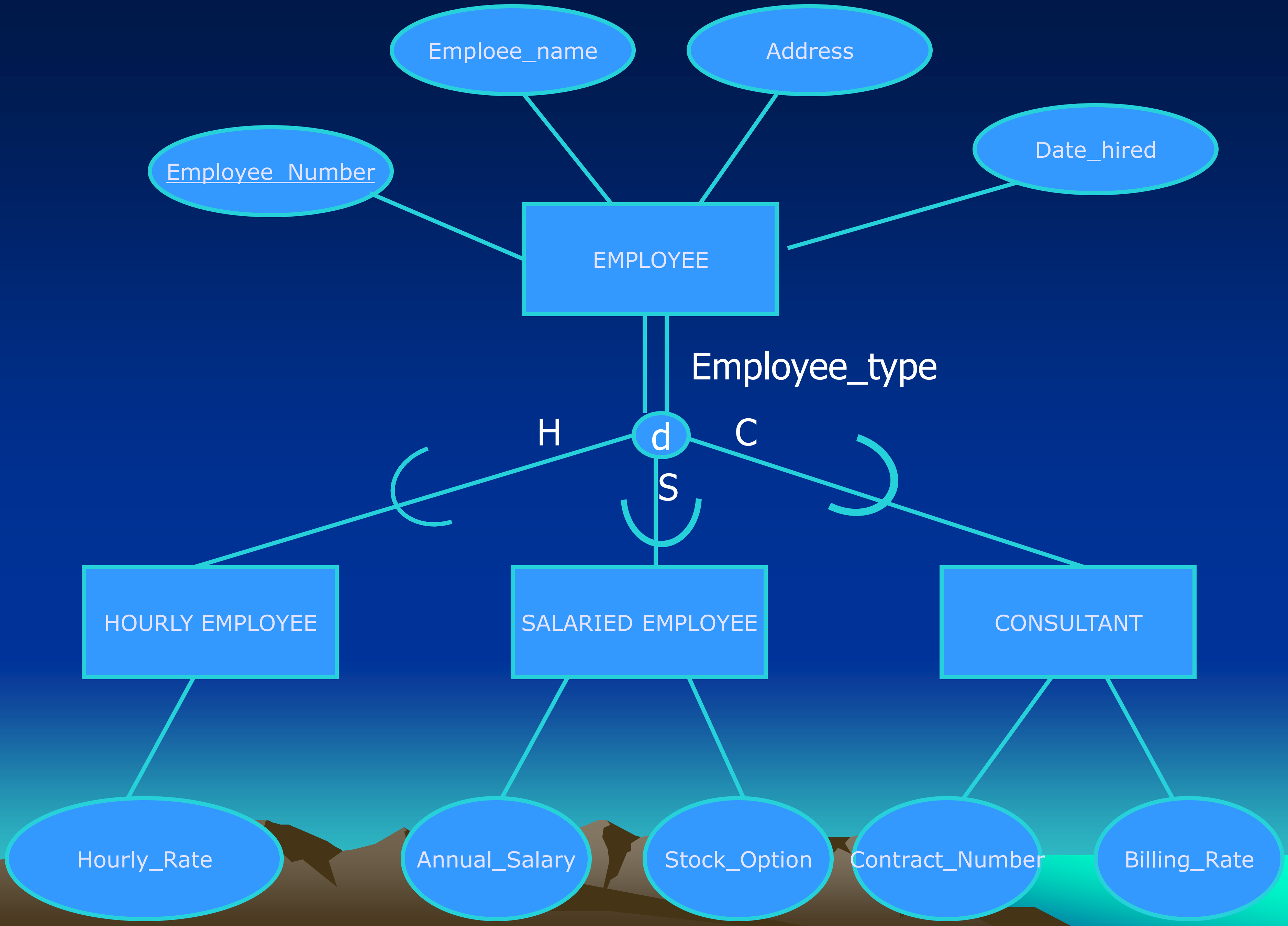
Treatment_Code	Description

STEP 7: MAP SUPERTYPE/SUBTYPE RELATIONSHIPS

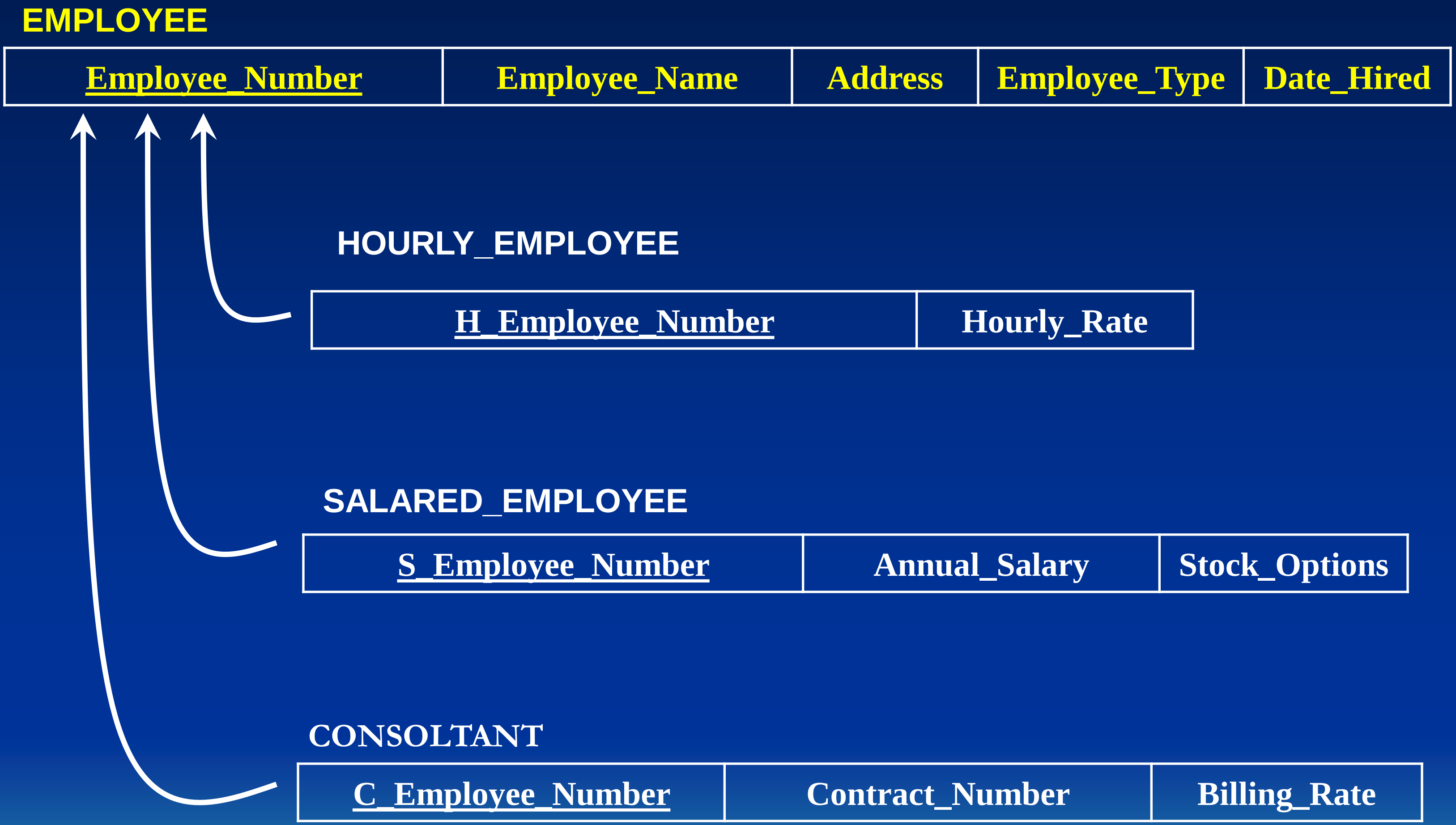
- **The relational data model does not yet directly support supertype / subtype relationships. Fortunately; there are various strategies that database designers can use to represent these relationships with the relational data model , for our purposes we use the following most commonly used strategy :**

1. **Create a separate relation for the supertype and each of its subtypes .**
2. **Assign to the relation created for the supertype the attributes that are commonly to all members of the supertype, including the primary key.**
3. **Assign to the relation for each subtype the primary key of the supertype ,and only these attributes that are unique to that subtypetype.**
4. **Assign one (or more) attributes of the supertype to function as the subtype discriminator.**

- An example of mapping this procedure is shown in the following two figures A and B
- Figure A shows the supertype EMPLOYEE with subtypes HOURLY EMPLOYEE, SALARED EMPLOYEE, and CONSOLTANT .
- The primary key of EMPLOYEE is Employee_Number ,and the attribute Employee_Type is the subtype discriminator


Emploee_name
Address
Employee_Number
Date_hired
EMPLOYEE
Employee_type
H
d
C
S
HOURLY EMPLOYEE
SALARIED EMPLOYEE
CONSULTANT
Hourly_Rate
Annual_Salary
Stock_Option
Contract_Number
Billing_Rate

- The result of mapping this diagram to relation using the above rule is shown in figure B.
- There is one relation for the supertype (EMPLOYEE) and one for each of the three subtypes .
- The primary key for each of the four relations is Employee_Number.
- A prefix is used to distinguish the name of the primary key for each subtype .
- For example ,S_Employee_Number is the name for primary key of the relation SALARED EMPLOYEE. Each of these attributes is a foreign key that references the supertype primary key, as indicated by arrows in the diagram.
- Each subtype relation contains only those attributes peculiar to the subtype.



EMPLOYEE

<u>Employee_Number</u>	Employee_Name	Address	Employee_Type	Date_Hired

HOURLY_EMPLOYEE

<u>H_Employee_Number</u>	Hourly_Rate

SALARED_EMPLOYEE

<u>S_Employee_Number</u>	Annual_Salary	Stock_Options

CONSOLTANT

<u>C_Employee_Number</u>	Contract_Number	Billing_Rate

- **For each subtype, a relation can be produced that contains *all* the attributes of that subtype(both specific and inherited)by using SQL command that joins the subtype with its supertype .**
- **For Example ,suppose that we want to display a table that contains all of the attributes for SALARED EMPLOYEE . The following command is used :**

```
SELECT *
FROM EMPLOYEE, SALARED EMPLOYEE
WHERE Employee_Number = S_Employee_Number;
```

Although you have not yet formally studied such commands, you can intuitively see that this command will join two tables and produce a large table that contains all the attributes from both tables.

END OF LECTURE 5